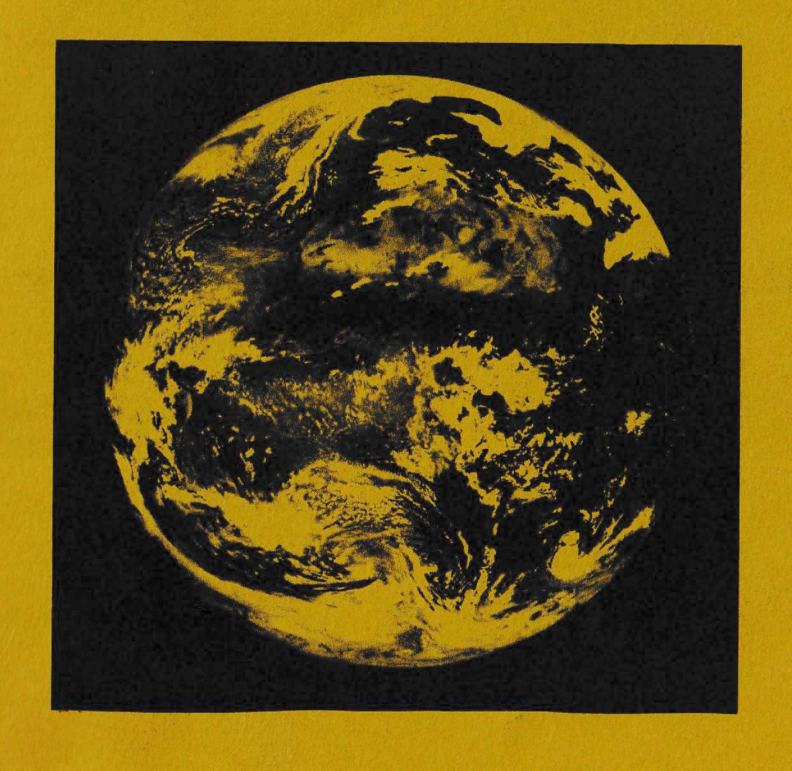
DER TI 99/4A WEITER AUF DEM VORMARSCH

TI 99/4A

* REPORT *

TITUS 6



EDITORIAL

Lieber Leser,
nach langem Warten liegt nun endlich die Weihnachtsausgabe des TI99/4A-REPORT vor Ihnen. Die Schar der TI-ler ist indessen kaum geschrumpft. Immer noch gibt es aktive User, die auch zu dieser Ausgabe ihren Teil beigetragen haben. An dieser Stelle möchte ich dann auch zur weiteren Mitarbeit auffordern, damit auch weiterhin Leben in dieser Zeit - schrift bleibt. Vielen Dank auch noch einmal an alle bisherigen Mitarbeiter.
Wer diese Ausgabe durchblättert, der findet neben den interessanten Artikeln auch noch die Angebote der Firmen auf dem TI-Markt.
Es ist wirklich unglaublich, aber das Angebot an neuen Produkten für den TI99/4A wird immer größer. Dabei sind die Preise mittlerweile auf ein günstiges Niveau eingependelt. Neu - heiten wie zum Beispiel eine 256 KRam Speicher - erweiterung oder die TI-Maus werden den Markt und das Interesse der User an dem Computer bestimmt auch noch 1986 kennzeichnen.
Der TI ist eben kein Computeropa, sondern ein echter Evergreen mit Pep. Tatsächlich ist das Gerät mit seinen neuen Möglichkeiten besser als manches "Neue". Daher möchte ich Allen auch weiterhin viel Spaß mit dem TI wünschen.

TEST



*** TEST HAGERA-ASSEMBLERKURS II ***

Bevor ich diesen Kurs in Händen hielt, bin ich oft gefragt worden, was bekommt man da für sein Geld, wie ist der Kurs aufgebaut, kann man damit gut arbeiten?

Nun, ich war angenehm überrascht. Schon das Äussere zeigt, dass der Autor sich sichtlich Mühe gegeben hat und Wert auf Qualität legt. Der Kurs besteht aus über 300 Seiten, die im DIN A5 Format in einem stabilen Ordner als Einzelblätter eingelegt sind. Ein farbiges Register trennt die einzelnen Abschnitte und erlaubt so einen schnellen Zugriff.

Leider bezieht sich der Kurs nur auf das Editor/Assembler-Paket und setzt daher einen voll ausgebauten Rechner voraus, mit Speichererweiterung, Diskettenlaufwerk und Editor/Assembler Modul. Auf einer mitgelieferten Diskette befinden sich die wichtigsten Beispielprogramme und ein Spiel, das im Kurs zur Übung beschrieben ist. Folgender Aufbau liegt dem Kurs zurunde: Der erste Abschnitt gibt eine Einführung in die Assemblersprache und bildet mit 120 Seiten den eigentlichen Lernkurs. Der Umgang mit dem Editor/Assembler wird als Erstes genau erklärt, so daß jemand, der zum erstenmal damit arbeitet gleich mit den Funktionen des Editor- und Assemblerprogramms vertraut gemacht wird. Die Beschreibung führt den User Schritt für Schritt durch die Menüs bis zur ersten Programmerstellung und dessen Start. Es ist erfreulich, daß der User von Anfang an durch Beispiele den Stoff vermittelt bekommt und sich nicht durch trockene Theorie quälen muß. Wo es geht, werden Vergleiche zum Basic hergestellt, um die Programme anschaulicher zu machen. Man lernt also durch das Ausprobieren von Beispielprogrammen, was sicher den Spass am Lernen erhält. Am Ende eines Kapitels soll man durch Änderungen dieser Beispielprogramme sein Wissen testen.

Welche Voraussetzungen werden vom Leser gefordert? Man sollte schon einmal in Basic programmiert haben, und Kenntnis von hexadezimalen Zahlen wäre zu empfehlen. Aber auch dies wird im Anhang kurz erklärt. Außerdem ist alles klar beschrieben und auf einfache Darstellungen reduziert.

Im zweiten Teil werden Problemlösungen zu oft gebrauchten Anwendungen aufgelistet und erklärt. Danach folgt die Lösung der Übungsaufgaben und das Listing eines Spiels, das vom User nach Vorschlägen erweitert werden soll, um das Wissen durch Anwendung zu vertiefen. Mehr als Nachschlagwerk folgt eine Erklärung der einzelnen Assemblerbefehle. Besonderheiten des TI-Assemblers, Erläuterungen zum Text und ein Inhaltsverzeichnis beenden das Werk.

Zusammengefaßt bietet dieser Kurs einen leicht verständlichen, interessant aufgemachten Einstieg in die Assemblerprogrammierung. Dabei stellt er keinen Anspruch auf Vollständigkeit oder will das Editor/Assemblerhandbuch ersetzen. Bei entsprechendem Bedarf soll in Zukunft eine Fortsetzung für Fortgeschrittene folgen.

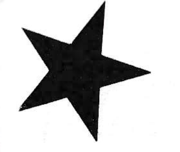
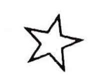
TI-Computer e.V.

Die Adresse des TI-Computer e.V. hat sich geändert. Der Herausgeber dieser Zeitschrift hat für die neue Amts - periode nicht mehr kandidiert. Leider lagen vom neuen Vorstand bis jetzt noch keine Informationen zur Veröffentlichung im TI99/4A-REPORT vor. Wir hoffen aber, daß in der nächsten Ausgabe näheres zu berichten ist.

Anfragen an die Redaktion, sowie die Einlieferung von Programmen und Artikeln (auf eigene Gefahr) für den TI99/4A-REPORT sollen nur noch an folgende Adresse erfolgen:

TI99/4A-REPORT
Hartmut Dirks
Reutershagweg 21
D - 5100 Aachen

Telefon:
0241 - 87 22 05

*** PROGRAMME FüR DAS MINIMEM ***

Gute Programme für das Mini Memory Modul sind Mangelware. Wer beherrscht schon die Assemblersprache und macht sich die Arbeit, mit dem Line by Line Assembler Programme zu entwerfen.

Die Resonanz auf unseren Aufruf, Programme für das Minimem an die Redaktion zu schicken, war außerordentlich gering. Uns ist es jedoch gelungen, sehr interessante Software zusammenzustellen, die aus professioneller Feder stammt.

Dank freundlicher Genehmigung von Herrn Mag. Karl Hagenbuchner dürfen wir hier einige der von ihm entwickelten Programme veröffentlichen.

Dabei haben wir uns für den Abdruck der Programme in Form von Basic-Programmen entschieden. Die Maschinenprogramme werden dabei über Pokebefehle in das Minimem übertragen. Dies ist die kürzeste Form der Eingabe und auch der Druckmöglichkeit von Assemblerprogrammen. Außerdem haben wir für den Anfang nur relativ kurze Programme, dafür aber Mehrere ausgewählt, in der Hoffnung, daß für jeden etwas interessantes dabei ist. Bei langen Programmen würden sich auch beim Abtippen schnell zu viele Tippfehler einschleichen, die dann sofort zum Programmabsturz führen können.

Fünf Programme folgen auf den nächsten Seiten. Sie erweitern das TI Basic um Funktionen, die bei kaum einem Computer zu finden sind, außer in einer wesentlich höheren Preisklasse.

1. MERGE

   Hiermit können Sie Basicprogramme in TI Basic miteinander verbinden. Das funktioniert sowohl mit Diskette, als auch mit Cassettenrecorder (geht normalerweise selbst in Extended Basic nicht).

2. GOTO/DATA

   Goto zum unmittelbaren Sprung, wie bekannt. Mit dem Databefehl wird der Restore-Befehl des Extended Basic nachvollzogen, mit Angabe einer Zeilennummer, ab der ein Read-Befehl Daten einlesen soll.

3. DELETE

   Löscht gewünschte Teile aus einem Basicprogramm.

4. BILDSCHIRM TOTAL

   Umfangreiche Programme zur Bildschirmgestaltung.

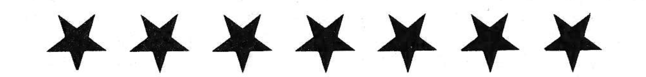
5. SMALL

   Mit diesem Programm kann der TI endlich ordentliche Kleinbuchstaben mit Umlauten auf den Bildschirm bringen.

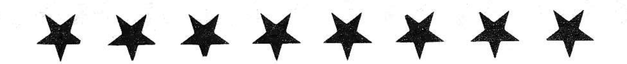
  




```
100 REM  MERGE-PROGRAMM FUER MINIMEM
110 REM  Dieses Programm erlaubt ein MERGE von Kasseten- bzw. Diskettenpr
n in TI-Basic                                                    ogramme
120 REM  Vorgang: (1) erstes Programm laden oder eingeben
130 REM           (2) CALL LINK("SAVE")
140 REM           (3) zweites Programm laden oder eingeben (Zeilennummern
ch !)                                                             unglei
150 REM           (4) CALL LINK("MERGE")
160 REM  Der Vorgang kann beliebig wiederholt werden!
170 REM  ================================================================
180 REM  (c) H.Reidlinger   Rev.Vers. K.Hagenbuchner    84/04
190 REM  ================================================================
200 DATA 2,10,117,164,195,160,131,48,195,224,131,50,131,143,18,18
210 DATA 206,142,206,143,198,160,131,112,6,14,200,14,131,112,200,14
220 DATA 131,48,200,14,131,50,4,224,131,62,4,91,2,10,117,164
230 DATA 195,186,22,4,2,0,25,0,4,32,96,80,195,250,193,218
240 DATA 6,74,6,74,131,143,18,246,131,160,131,112,20,243,195,96
250 DATA 131,50,195,13,194,96,131,48,130,77,18,95,2,8,116,168
260 DATA 192,72,2,2,0,2,192,14,4,32,96,40,193,88,2,3
270 DATA 0,3,2,4,0,4,192,13,96,3,4,32,96,48,129,88
280 DATA 21,4,2,0,26,0,4,32,96,80,4,197,192,143,96,142
290 DATA 5,130,129,2,17,44,2,130,0,252,18,2,2,2,0,252
300 DATA 192,14,193,142,163,130,161,66,4,32,96,48,131,70,18,24
310 DATA 6,6,192,6,6,192,216,0,140,2,6,192,216,0,140,2
320 DATA 16,0,208,96,136,0,160,2,2,96,64,0,6,192,216,0
330 DATA 140,2,6,192,216,0,140,2,16,0,216,1,140,0,16,230
340 DATA 192,72,192,13,5,128,4,32,96,40,163,66,16,207,192,9
350 DATA 2,2,0,2,192,72,5,192,131,0,20,7,4,32,96,48
360 DATA 166,5,4,32,96,40,160,4,16,247,200,13,131,50,200,7
370 DATA 131,112,4,218,216,26,131,124,4,91,200,58,131,48,200,26
380 DATA 131,50,6,74,16,244
390 FOR ADR=30122 TO 30415
400 READ POKEVAL
410 CALL LOAD(ADR,POKEVAL)
420 NEXT ADR
430 CALL LOAD(28702,127,240)
440 CALL LOAD(32752,77,69,82,71,69,32,117,214)
450 CALL LOAD(32760,83,65,86,69,32,32,117,170)
```



```
10 REM  DATA-LIST: mit diesem Programm wird ermöglicht:
20 REM       CALL LINK("GOTO",Zeilennr.)
30 REM       CALL LINK("DATA",Zeilen-,DATA-Nummer)
40 REM  Bei "GOTO" wird ein unmittelbarer Sprung erzwungen
50 REM      "DATA" stellt die Zeiger auf ein bestimmtes DATA
60 REM             Dieses wird mit der nächsten READ-Anweisung gelesen.
70 REM  ================================================================
80 REM  (c) H.Reidlinger    Rev.Pokeliste K.Hagenbuchner  84/4
90 REM  ================================================================
100 DATA 194,139,2,1,0,1,6,160,113,66,192,224,131,50,6,3
```



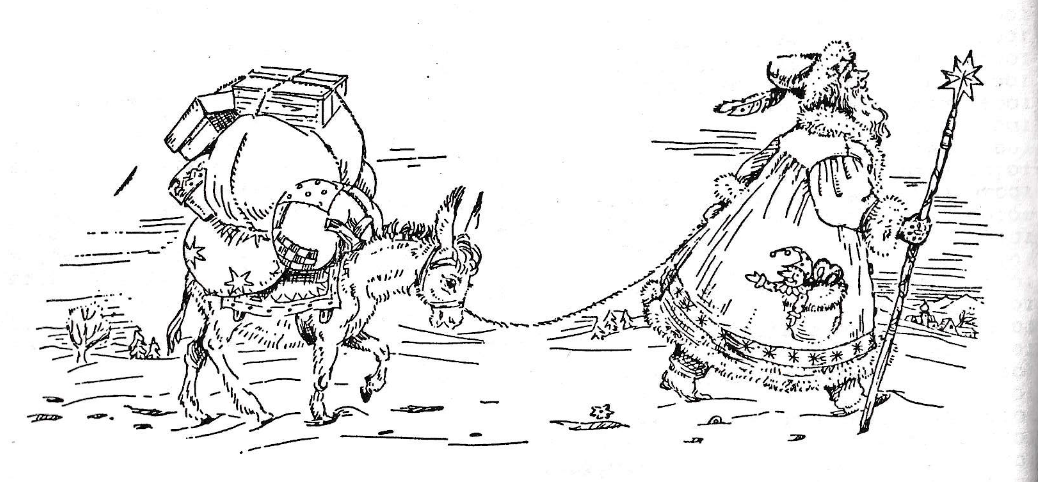
```
100 REM  BILDSCHIRM-TOTAL: Maschinensprachenprogramme zur
110 REM                    Bildschirmgestaltung
120 REM
130 REM  (c) Mag.Karl Hagenbuchner,Haidfeldstr. 2, A-4050 Traun
140 REM  Diese Pokeliste erzeugt ein Maschinenprogramm, mit dessen Hilfe
150 REM  folgende Routinen ermöglicht werden:
160 REM   CALL LINK("LINKS")       scrollt um eine Spalte nach links
170 REM   CALL LINK("RECHTS")      scrollt um eine reihe nach rechts
180 REM   CALL LINK("DOWN")        scrollt um eine Zeile herunter
190 REM   CALL LINK("HIDE")        speichert einen Bildschirm
200 REM   CALL LINK("SEEK")        bringt gespeicherten Bildschirm
210 REM   CALL LINK("WINDOW",Z1,Z2,S1,S2,DIR)
220 REM                            scrollt einen Bildschirmabschnitt zw. Zeile Z1
230 REM                            und Zeile Z2, Spalte1 und Spalte2 hinauf (DIR=0)
240 REM                            oder hinunter (DIR=1)
250 REM   CALL LINK("MASKE",Z1,Z2,S1,S2,DIR)
260 REM                            scrollt den Schirm bei ruhender Maske, begrenzt
270 REM                            durch Zeile1, Zeile 2, Spalte1, Spalte2
280 REM                            nach links (DIR=0), rechts (DIR=1),
290 REM                            hinunter (DIR=2) und hinauf (DIR=3)
300 REM    CALL LINK("REVERS",ASCII1,ASCII2)
310 REM                            setzt einzelne Charakters (ASCII 1) invers und
320 REM                            definiert damit einen zweiten Charakters
330 REM                            Rückbezug ist möglich: ASCII 1 = ASCII 2
340 REM  ***********************************************************************
*
350 REM  Dieses Programm läuft nur mit MINIMEM !
360 REM  ***********************************************************************
*
370 CALL INIT
380 DATA 194,139,6,160,121,162,5,129,5,133,4,32,96,40,2,32,0,31,192
390 DATA 68,4,32,96,36,5,128,2,37,0,32,192,69,6,3,22,243,4,90,194
400 DATA 139,6,160,121,162,5,128,4,32,96,40,6,0,192,68,4,32,96,36,2
410 DATA 32,0,32,2,37,0,32,192,69,6,3,22,242,4,90,4,192,2,1,118,56
420 DATA 2,2,3,0,4,32,96,48,193,65,2,4,128,0,2,3,0,24,2,2,0,31,4
430 DATA 91,4,192,2,1,118,56,2,2,2,224,4,32,96,48,2,32,0,32,4,32,96
440 DATA 40,2,1,128,0,2,2,0,32,4,192,4,32,96,36,5,128,6,2,22,251,4
450 DATA 91,194,139,6,160,122,36,4,32,96,48,6,160,122,48,4,32,96,48,6
460 DATA 160,122,62,4,32,96,48,4,90,194,139,6,160,122,36,4,32,96,40,6
470 DATA 160,122,48,4,32,96,40,6,160,122,62,4,32,96,40,4,90,4,192,2
480 DATA 1,113,24,2,2,3,0,4,91,2,0,3,0,2,1,116,24,2,2,0,32,4,91,2
490 DATA 0,4,0,2,1,116,56,2,2,2,0,4,91
500 DATA 4,32,96,68,4,32,96,28,18,0,142,224,131,74,17,4,142,224,131
510 DATA 74,21,1,4,91,2,0,19,0,4,32,96,80,194,139,4,192,2,1,0,1,6
520 DATA 160,122,76,0,1,0,23,193,160,131,74,5,129,6,160,122,76,0,2,0
530 DATA 24,129,160,131,74,17,234,193,224,131,74,5,129,6,160,122,76,0
540 DATA 1,0,31,194,32,131,74,5,129,6,160,122,76,0,2,0,32,130,32,131
550 DATA 74,17,217,194,96,131,74,5,129,6,160,122,76,0,0,0,3,195,96
560 DATA 131,74,4,90,195,11,6,160,122,108,6,6,10,86,161,136,5,134,6,7
570 DATA 10,87,161,200,5,135,5,137,98,72,2,141,0,0,19,37,2,141,0,1
580 DATA 22,188,192,7,2,32,0,32,2,1,121,56,192,137,2,32,255,192,4,32
590 DATA 96,48,129,128,19,5,2,32,0,32,4,32,96,40,16,245,2,32,0,32,4
600 DATA 32,96,40,2,32,255,224,2,15,128,0,220,79,6,2,22,253,2,1,121
610 DATA 56,192,137,4,32,96,40,4,92,192,6,2,32,255,224,2,1,121,56,192
620 DATA 137,2,32,0,64,4,32,96,48,2,32,255,224,129,192,17,9,19,1,4
630 DATA 92,195,65,195,130,2,15,128,0,223,79,6,14,22,253,4,32,96,40,16,236
640 DATA 194,139,4,192,2,1,0,1,6,160,123,192,192,224,131,74,5,129,6
650 DATA 160,123,192,193,32,131,74,2,0,3,248,2,5,0,31,2,32,0,8,6,3
```



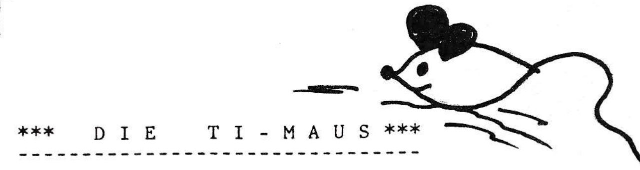
```
660 DATA 129,67,22,251,2,1,121,56,2,2,0,8,4,32,96,48,2,6,121,56,2,3
670 DATA 0,4,5,22,6,54,6,3,22,252,2,0,3,248,2,32,0,8,6,4,129,68,22
680 DATA 251,4,32,96,40,4,90,4,32,96,68,4,32,96,28,18,0,4,91
690 DATA 195,11,6,160,122,108,195,77,19,7,2,141,0,1,19,7,2,141,0,2
700 DATA 19,7,16,11,6,160,121,88,16,34,6,160,121,126,16,31,6,160,121
710 DATA 162,6,160,121,192,16,26,6,160,121,162,2,0,0,32,2,2,0,32,4
720 DATA 32,96,48,2,32,255,224,4,32,96,40,2,32,0,64,2,128,3,0,17,245
730 DATA 2,0,2,224,2,1,128,0,4,32,96,36,5,128,6,2,22,251,6,160,124
740 DATA 162,195,77,19,17,2,141,0,1,19,25,2,141,0,2,19,34,2,1,128,0
750 DATA 192,6,2,32,255,224,4,32,96,36,5,128,6,2,22,251,4,92,192,6,6
760 DATA 0,2,1,128,0,4,32,96,36,2,32,0,32,6,14,22,250,4,92,192,6,5
770 DATA 128,160,2,2,1,128,0,4,32,96,36,2,32,0,32,6,14,22,250,4,92
780 DATA 192,6,5,142,2,32,0,32,6,14,22,252,2,1,128,0,4,32,96,35,5
790 DATA 128,6,2,22,251,4,92,6,6,6,7,6,8,6,9,193,70,10,86,161,136
800 DATA 192,6,2,1,118,56,160,64,97,197,5,135,195,135,192,137,96,136,5
810 DATA 130,4,32,96,40,2,32,0,32,2,33,0,32,6,7,22,248,4,91,-1
820 I=31064
830 READ POKE
840 IF POKE=-1 THEN 880
850 CALL LOAD(I,POKE)
860 I=I+1
870 GOTO 830
880 CALL LOAD(28700,123,100,127,192)
890 I=32704
900 DATA 76,73,78,75,83,32,121,88,82,69,67,72,84,83,121,126,68,79,87
910 DATA 78,32,32,121,192,72,73,68,69,32,32,121,236,83,69,69,75,32,32
920 DATA 122,8,87,73,78,68,79,87,122,198,73,78,86,69,82,83,123,100
930 DATA 77,65,83,75,69,32,123,204,-1
940 READ POKE
950 IF POKE=-1 THEN 990
960 CALL LOAD(I,POKE)
970 I=I+1
980 GOTO 940
990 END
```

```
10000 REM  Kleinbuchstaben und Umlaute angepasst an Drucker GP100A S/GP700A
10010 REM  Pokeliste fuer MINIMEM
10020 REM  Das Programm nimmt 306 Byte ab >7118 in Anspruch
10030 DATA 2,0,5,216,2,1,113,42,2,2,1,40,4,32,96,40,4,91
10040 DATA 0,40,0,56,68,124,68,68,0,40,0,56,68,68,68,56,0,40,0,68,68,68,68,56
10050 DATA 0,8,20,34,0,0,0,0,0,0,0,0,0,255,0,0,16,8,4,0,0,0
10060 DATA 0,0,0,56,8,56,72,52,0,64,64,88,100,68,68,120,0,0,0,60,64,64,64,60
10070 DATA 0,4,4,52,76,68,68,60,0,0,0,56,68,120,64,56,0,24,36,32,112,32,32,32
10080 DATA 0,0,56,68,68,60,4,56,0,64,64,88,100,68,68,68,0,16,0,48,16,16,16,56
10090 DATA 0,8,0,8,8,8,40,16,0,64,64,72,80,96,80,72,0,48,16,16,16,16,16,24
10100 DATA 0,0,0,104,84,84,84,68,0,0,0,88,36,36,36,36,0,0,0,56,68,68,68,56
10110 DATA 0,0,112,72,72,112,64,64,0,0,56,72,72,56,8,8,0,0,0,88,100,64,64,64
10120 DATA 0,0,0,56,64,56,4,120,0,32,32,120,32,32,36,24,0,0,0,72,72,72,72,52
10130 DATA 0,0,0,68,68,40,40,16,0,0,0,68,68,84,84,40,0,0,0,68,40,16,40,68
10140 DATA 0,0,0,136,136,56,8,112,0,0,0,124,8,16,32,124,0,40,0,56,72,72,72,60
10150 DATA 0,40,0,56,72,72,72,56,0,40,0,68,68,68,68,52,24,36,36,40,36,36,36,72
10160 CALL INIT
10170 FOR ADRESSE=28952 TO 29257
10180 READ POKE
10190 CALL LOAD(ADRESSE,POKE)
10200 NEXT ADRESSE
10210 CALL LOAD(28700,114,74,127,248)
10220 CALL LOAD(32760,83,77,65,76,76,32,113,24)
10230 REM  AUFRUF MIT CALL LINK("SMALL")
```

```
10 REM  DELETE-ZEILENNUMMER MIT MINIMEM
20 REM  Dieses Programm löscht aus Programme gewünschte Teile
30 REM  Aufruf: CALL LINK("DEL",ZNR1,ZNR2)
40 REM                ZNR1... ab dieser Zeilennummer soll bis
50 REM                ZNR2 geloescht werden
60 REM  Die Zeilennummern muessen nicht existent sein, es werden die naechsthoeh
eren genommen
70 REM  ====================================================================
80 REM  (c) H.Reidlinger Rev.Vers. K.Hagenbuchner      84/4
90 REM  ====================================================================
100 DATA 4,32,96,68,4,32,96,28,18,0,193,160,131,74,21,4
110 DATA 2,0,26,0,4,32,96,80,4,91,194,139,4,192,2,1
120 DATA 0,1,2,2,0,4,6,160,126,76,193,70,5,129,6,160
130 DATA 126,76,129,133,21,237,193,224,131,50,194,32,131,48,130,7
140 DATA 22,4,2,0,25,0,4,32,96,80,192,200,195,224,131,112
150 DATA 194,71,6,160,104,254,129,129,18,4,160,194,129,195,17,249
160 DATA 16,215,129,65,17,213,193,131,195,6,5,195,163,2,6,160
170 DATA 104,254,6,1,195,65,192,193,6,160,105,58,9,129,5,129
180 DATA 195,129,6,3,193,3,161,14,129,195,17,69,131,196,21,67
190 DATA 130,67,22,2,194,68,16,35,6,160,105,58,6,160,105,76
200 DATA 6,3,6,4,130,67,21,248,10,36,9,36,194,68,163,76
210 DATA 192,200,193,8,5,195,5,196,131,3,21,5,129,131,17,5
220 DATA 192,204,193,12,16,247,129,195,21,10,6,160,104,254,131,65
230 DATA 21,3,160,78,6,160,105,26,160,194,161,2,16,237,192,204
240 DATA 129,204,21,4,6,160,104,254,129,65,20,191,200,9,131,50
250 DATA 193,9,6,4,192,199,6,3,129,131,17,4,131,3,20,4
260 DATA 192,198,16,7,130,3,17,10,6,160,104,254,6,160,105,26
270 DATA 6,68,6,67,16,241,2,0,38,0,16,232,5,196,10,36
280 DATA 9,36,19,2,131,196,17,1,193,15,200,4,131,48,4,207
290 DATA 200,15,131,62,192,0,19,2,4,32,96,80,4,90
300 FOR ADR=32332 TO 32649
310 READ POKEVAL
320 CALL LOAD(ADR,POKEVAL)
330 NEXT ADR
340 CALL LOAD(28700,127,138,127,248)
350 CALL LOAD(32760,68,69,76,32,32,32,126,102)
```



TEST



*** D I E   T I - M A U S ***

Der TI ist noch lange nicht am Ende. Erst in letzter Zeit erscheinen immer mehr neue Software- und Hardwareprodukte. Angefangen von neuen Programmiersprachen wie Forth, Basic-Compilern und billigen Assemblerpaketen, bis zu anspruchsvollen Grafikpaketen wie das Extended Basic II. Da inzwischen das Betriebssystem etwas durchsichtiger geworden ist, und man auch ohne große Investitionen Assemblerprogramme schreiben kann, können neue Ideen besser verwirklicht werden, und der TI kann zeigen, was alles in ihm steckt.

Neben Hardwareerweiterungen wie dem Mini-Assembler gibt es inzwischen auch ein Grafik-Tablett, ein Modul zum Steuern beliebiger Anlagen (z.B. Eisenbahn), Eprommer, etc.

Neu ist ein in letzter Zeit immer bekannter werdendes Eingabegerät: Die Maus. Publik gemacht durch Appels Computer Lisa, Basis des Nachfolgemodells Macintosh und in Zukunft auch in der Grundausstattung von Ataris Neuestem, dem 520 ST.

Was ist das Besondere an dieser Technik?

Bisher wird der User anhand von Menüs durch das Programm geführt. Bei komplizierter Programmstruktur wandert man so von Menü zu Menü und verliert dabei leicht die Übersicht. Dazu sind oftmals die anzuwählenden Punkte unklar beschrieben.

Die neue Technik wendet bei der Benutzerführung mehr Grafik an. Die Programmteile oder Funktionen werden, wenn möglich durch grafische Symbole dargestellt. Mit der Maus, die über eine ebene Fläche bewegt wird, steuert man den Cursor über dieses Symbol und läßt durch Tastendruck die entsprechende Funktion ausführen. Die Grundsymbole bleiben während des ganzen Programms auf dem Bildschirm. Auf einem Fenster, d.h. einem Ausschnitt auf dem Bildschirm, wird die Funktion ausgeführt. Dadurch bleibt das Programm übersichtlich, und durch die Grafik sind die Befehle oft eindeutiger definiert.

Wie sieht die Maus nun aus?

Die Lieferung umfaßt eine Diskette oder Cassette mit der Software zur Steuerung und einigen Demonstrationsprogrammen. Die Maus ist aus einem rechteckigen, handflächengroßen Gehäuse mit zwei großen Tastern aufgebaut, in dem sich eine Kugel befindet. Gleitet man mit diesem Gerät nun über eine ebene Fläche, so dreht sich die Kugel entsprechend und gibt Impulse an den Rechner weiter. Dies geschieht über ein Interface, über das die Maus einfach mit dem Joystickanschluß verbunden wird. Interface und die Electronic der Maus werden über ein kleines mitgeliefertes Steckernetzteil mit Spannung versorgt. Die Impulse werden durch die Steuersoftware ausgewertet. Das sind kleine Assemblerprogramme, die mit CALL LINK in das Basic-Programm eingebunden werden können (z.B. anstelle von CALL JOYST).

Die Steuerprogramme sind unterteilt in zwei Abfrageversionen. Bei Version 1 CALL LINK("MOUSEEO") wird die Ausführung des Basic-Programms solange unterbrochen, bis die Maustaste gedrückt wird. Bei Version 2 (CALL LINK("MOUSE1")) handelt es sich um eine Interrupt-Routine. Das heißt, daß immer, wenn ein Interrupt vom Basicinterpreter zugelassen wird, die Position der Maus abgefragt und der Cursor gesetzt wird. Um diesen Modus zu verlassen wird CALL LINK("MCLR") verwendet. Als Cursor wird immer Sprite Nr.1 verwendet. Dieser Sprite muß daher im Basic- bzw. Assemblerprogramm definiert werden. Über CALL PEEK und CALL LOAD kann jederzeit die Cursorposition abgefragt oder gesetzt werden. Dabei ist es möglich, ein Fenster zu definieren, in dem sich der Cursor bewegen darf. Um einen definierten Anfangspunkt zu haben, kann man über den zweiten Taster der Maus den Cursor an eine bestimmte Anfangsposition bewegen.

Diese Assemblersoftware muß nur einmal nach dem Einschalten des Rechners geladen werden, und befindet sich dann im unteren Teil der Speichererweiterung.

Wenn man die Steuerung in ein eigenes Assemblerprogramm einbinden will, so steht auf der Diskettenversion auch der Quellcode zur Verfügung.

Zur Demonstration dient das Programm CALCULATOR. Auf dem Bildschirm erscheint ein kleiner Taschenrechner, und eine Hand, die mit der Maus gesteuert wird. Ein Klicken mit der Maustaste führt die entsprechende Funktion aus. Dieses Beispiel zeigt überzeugend die Vorteile dieser Eingabemethode. Die Bedienung ist jedem sofort klar und eindeutig. Eine Erklärung oder ein kompliziertes Menü entfällt. Alle Funktionen des Programms werden über ein einziges Bild erreicht.

Das ist aber nicht die einzigste Anwendungsmöglichkeit. Die Maus bietet sich genauso zur komfortablen Spielsteuerung an. Das zeigt eindrucksvoll das Spiel BREAKOUT (ebenfalls mit in der Lieferung). Es ist den meisten als Tennis oder Pelota aus den Anfängen der Videospiele bekannt. Der Tennisschläger kann hierbei im unteren Teil des Spielfeldes in alle Richtungen bewegt werden. Da die Bewegung der Maus sowohl in der Geschwindigkeit, als auch in dem zurückgelegten Weg mit dem Schläger übereinstimmt, ist eine sehr gefühlvolle und genaue Steuerung möglich. Sie werden schnellstens Ihren Joystick ausmustern, und nur noch mit der Maus spielen und programmieren.

Doch folgendes muß erwähnt werden. Die Maus kann den Joystick nicht ersetzen. Programme, die für den Joystick geschrieben worden sind, können ohne Änderung nicht für die Maus übernommen werden. Daher bleibt nur zu hoffen, daß noch recht viel Software für diese gelungene Erweiterung des TI's in Zukunft auf den Markt kommen wird.

VIER GEWINNT

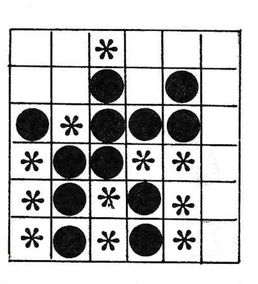
*** TI - BASIC PROGRAMM ***

Auch für den User, der nur eine Konsole mit Cassettenrecorder hat, haben wir das Listing zu einem interessanten, wohlbekannten Spiel abgedruckt: "Vier Gewinnt".

Es ist Sinn des Spiels, vier Punkte aneinanderzureihen, dabei ist egal, ob diagonal, vertikal oder horizontal.

Viel Spaß !

```
100 CALL CHAR(128,"1818181818181818")
110 CALL CHAR(129,"000000FFFF000000")
120 CALL CHAR(39,"FFFFFFFFFFFFFFFF")
130 CALL CLEAR
140 PRINT "   **************          *VIER GEWINNT*          **********
****"
150 PRINT
160 PRINT "by $HANS JOACHIM GERSTEIN"
170 PRINT
180 PRINT
190 PRINT
200 FOR A=16 TO 4 STEP -1
210 CALL SCREEN(A)
220 CALL SOUND(500,A*100,4)
230 FOR B=1 TO 100
240 NEXT B
250 NEXT A
260 CALL CLEAR
265 RESTORE
270 INPUT "E FUER ERKLAERUNG DES SPIELS":E$
280 IF E$="E" THEN 1060
290 INPUT "FARBE :":R
300 CALL CLEAR
310 PRINT TAB(4);"1  2  3  4  5  6  7"
315 FOR V=22 TO 3 STEP -3
320 CALL HCHAR(V,6,129,20)
330 NEXT V
340 A=21
350 B=21
360 C=21
370 D=21
380 E=21
390 F=21
400 G=21
410 READ S
415 IF S=1 THEN 980
420 CALL SCREEN(R)
430 CALL SOUND(200,440,2)
440 FOR Q=1 TO 30
450 NEXT Q
460 CALL KEY(0,KEY,STATUS)
470 IF KEY=49 THEN 560
480 IF KEY=50 THEN 620
490 IF KEY=51 THEN 680
500 IF KEY=52 THEN 740
510 IF KEY=53 THEN 800
520 IF KEY=54 THEN 860
530 IF KEY=55 THEN 920
540 IF KEY=83 THEN 980
```




```
550 GOTO 460
560 IF A<5 THEN 460
570 X=6
580 CALL HCHAR(A,X,S,2)
590 CALL HCHAR(A-1,X,S,2)
600 A=A-3
610 GOTO 410
620 IF B<5 THEN 460
630 X=9
640 CALL HCHAR(B,X,S,2)
650 CALL HCHAR(B-1,X,S,2)
660 B=B-3
670 GOTO 410
680 IF C<5 THEN 460
690 X=12
700 CALL HCHAR(C,X,S,2)
710 CALL HCHAR(C-1,X,S,2)
720 C=C-3
730 GOTO 410
740 IF D<5 THEN 460
750 X=15
760 CALL HCHAR(D,X,S,2)
770 CALL HCHAR(D-1,X,S,2)
780 D=D-3
790 GOTO 410
800 IF E<5 THEN 460
810 X=18
820 CALL HCHAR(E,X,S,2)
830 CALL HCHAR(E-1,X,S,2)
840 E=E-3
850 GOTO 410
860 IF F<5 THEN 460
870 X=21
880 CALL HCHAR(F,X,S,2)
890 CALL HCHAR(F-1,X,S,2)
900 F=F-3
910 GOTO 410
920 IF G<5 THEN 460
930 X=24
940 CALL HCHAR(G,X,S,2)
950 CALL HCHAR(G-1,X,S,2)
960 G=G-3
970 GOTO 410
980 PRINT
990 PRINT "ICH HOFFE ES HAT DIR":"SPASS GEMACHT":"WOLLEN WIR WEITERSPIELEN(J-N)"
1000 INPUT M$
1010 IF M$="J" THEN 260
1020 STOP
1030 DATA 39,79,39,79,39,79,39,79,39,79,39,79,39,79,39,79,39,79,39,79,39,79,39,7
9
1040 DATA 39,79,39,79,39,79,39,79,39,79,39,79,39,79,39,79,39,79
1050 DATA 1
1060 PRINT "SINN DIESES SPIELS IST ES   VIER PUNKTE DIAGONAL,        VERTIKAL ODE
R HORIZONTAL    "
1070 PRINT "ANEINANDER ZU REIHEN.DER,   DEM DAS GELINGT,HAT GE-     WONNEN."
1080 PRINT "WENN DU DAS SPIEL ABBRECHEN WILLST ODER EINER GEWONNEN HAT DRUECKE S
"
1090 PRINT "WENN DU MIT DEM LESEN FERTIG BIST,DRUECKE F"
1100 INPUT F$
1110 GOTO 290
```



*** TEST 32K-SPEICHERERWEITERUNG ***

Neu auf dem Markt ist eine 32K-Erweiterung der Firma Electronic Hardware Service Laloire. Sie erweitert das bisherige Angebot an Speichererweiterungen um eine interessante Variante.

Die Erweiterung wird rechts seitlich an der Konsole angeschlossen. Sie ist ca. 7cm breit, aus dem gleichen schwarzen Plastikmaterial der Konsole und schließt in der Tiefe mit dem Computer ab. Der Busstecker ist in der Höhe angepaßt, so daß eine stabile Verbindung zur Konsole besteht, besonders wichtig beim Anschluß weiterer Peripherie an den durchgeführten Bus. Ein Anschluß z.B. einer Peripheriebox oder Schnittstelle ist also problemlos möglich. Ein Kontroll-LED zeigt die Betriebsbereitschaft an.

Da die Speichererweiterung in hochwertiger CMOS-Technik aufgebaut ist, kann der Inhalt über einen eingebauten Akkumulator auch bei abgeschalteter Konsole auf Dauer erhalten bleiben. Der Akkumulator wird im Betrieb automatisch nachgeladen. Daher ergibt sich eine hohe Lebensdauer (keine Batterie wie z.B. im Minimem).

Die Lieferung schließt ebenfalls die notwendige Software ein, um die Batteriepufferung auch nutzen zu können.

Erst mit einer 32K-Erweiterung ist in Verbindung mit Extended Basic Assemblersprache möglich. Assemblerprogramme bleiben bei Pufferung auch nach dem Ausschalten der Konsole erhalten. Somit stehen die Programme, bzw. Erweiterungen des Basics um z.B. Befehle zur hochauflösenden Grafik, direkt nach dem Einschalten der Konsole wieder zur Verfügung. Ein Laden von Cassette oder Diskette entfällt.

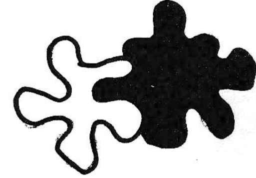
Bei Basicprogrammen ist das nicht ganz so einfach. Informationen, die das Betriebssystem in der Konsole abgelegt hat, sind natürlich verloren gegangen. Dieses Manko wird jedoch dank der mitgelieferten Software beseitigt. Nach einmaliger Eingabe von CALL LINK("SAVE") vor dem Ausschalten der Konsole ist das Basic-Programm nach neuer Inbetriebnahme durch CALL LINK("OLD") in Sekundenbruchteilen wieder zur Verfügung.

Weitere Software zu diesem Thema ist gegen einen geringen Aufpreis ebenfalls lieferbar.

Neben der bekannten Erweiterung Extended Basic II von Herrn Kull (hochauflösende Grafik, Kleinbuchstaben mit echten Unterlängen, permanente Uhr, Zeitsteuerung ...), gibt es ein umfangreiches Assemblerpaket. Diese Software erlaubt für wenig Geld die Erstellung von Assemblerprogrammen, deren Speicherung und Test. Es wird ein Line by Line, sowie ein Zwei-Weg Assembler geboten. Das Paket ist insbesondere für den User ohne Diskettenlaufwerk interessant, der das Editor/Assemblerpaket nicht nutzen kann.

Zusammenfassend kann man sagen, daß hier für wenig Geld eine lohnende Erweiterung zum TI geboten wird, die viele interessante Anwendungen erst möglich macht.





\>>> T I P S   U N D   T R I C K S <<<

Folgender Beitrag stammt vom Leser Roman Romanowski. Er ist seit einem halben Jahr Besitzer des Adventures "RETURN TO PIRATE'S ISLE". Er hat das Rätsel zwar insgesamt noch nicht lösen können, aber er hat es immerhin geschafft, das Schiff zu fahren (was ja eigentlich die Hauptaufgabe ist).

In den nachfolgenden Zeilen wird beschrieben, wie er das geschafft hat. Diejenigen, die es lieber selbst lösen wollen, sollten die nächsten Zeilen nicht lesen. Natürlich sind auf der Insel noch andere Geheimnisse verborgen, die hier jedoch nicht verraten werden!

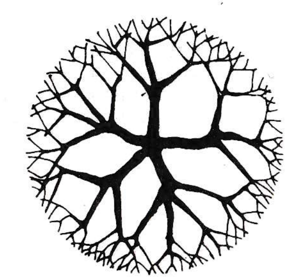
Also, wenn Sie den Piraten nicht geweckt, alle Schätze geborgen und auf das Schiff gebracht haben, können Sie alle Vorbereitungen zum Auslaufen des Schiffes treffen. Dies geht so:

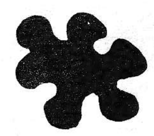
In der Kabine ist ein Schalter versteckt, den Sie finden müssen. Geben Sie "LOOK CEILING" ein. Den Schalter dürfen Sie aber noch nicht betätigen.

Um die wichtige Karte zu erhalten müssen Sie das Bild, auf dem ein Pirat abgebildet ist besitzen. Gehen Sie dann in den Maschinenraum und holen Sie den Schraubenzieher von der Maschine, indem Sie "GO ENGINE" und danach "LOOK ENGINE" eingeben. Danach "REMOVE FRAME" und "LOOK PICTURE". Jetzt besitzen Sie die Karte.

In dem Maschinenraum befinden sich Batterien und "GAS". Diese beiden Dinge dürfen nie auf "0" stehen. Drücken Sie nun den Knopf in der Kabine und warten Sie bis die Batterien unter "7 AMPS" stehen. Immer wenn Sie sich bewegen wird ein Ampere abgezogen. Wann die Ampere unter 7 sind erfahren Sie, wenn Sie "LOOK BATTERIES" eingeben. Wenn Sie im Besitz des Schraubenziehers sind, können Sie "REMOVE BLADE" eingeben, nachdem Sie "LOOK FAN" eingegeben hatten. Nun drücken Sie noch einmal den Knopf, gehen in den Maschinenraum und tippen "START ENGINE" ein und sofort danach "STOP ENGINE". Nun nehmen Sie die Klinge ("BLADE") und führen folgende Befehle durch:

"GO UP"
"GO UP"
"GO DOCK"
"GO NORTH"
"DIG"
"CLIMB HILL"
"JUMP UP"
"DROP RUM"
"WAKE PIRATE"
"JUMP LEDGE"





Gehen Sie nun wieder in das Schiff und danach in den Maschinenraum. Dort angelangt tippen Sie "START ENGINE" ein. Gehen Sie nun zurück in die Kabine und geben "DRIVE SHIP" ein. Es ist dann alles dunkel, wenn Sie aber den Knopf drücken wird es hell.

Zum Schluß noch eine Frage an andere "RETURN TO PIRATE'S ISLE"-Besitzer: Wie kann man den Wecker finden ? Wer die Antwort kennt, kann sich an die Radaktion des Titus wenden. Die Informationen werden dann weitergegeben, und vielleicht kommt es über die Vereinszeitschrift zu einem Informationsaustausch.

Roman Romanowski

Im Titus 4 (32K-Ram) haben sich leider ein paar Fehler im Schaltplan durch Übertragung einer Skizze in einen "ordentlichen Schaltplan" eingeschlichen. Hier die erforderlichen Korrekturen:

1. Pin 27 und 22 am 6264 Ram wurden vertauscht (Pin 27=WE, Pin 22=OE).

2. Pin 1 vom Datenbustreiber 74245 muß mit dem invertierten DBIN-Signal (Ausgang 74240) verbunden werden.

3. Die CS-Leitungen (Pin 20 von 6264) müssen mit je 1kOhm gegen +B gelegt werden.

4. Beim IC 74138 wurde ein TI-Typ verwendet, da Probleme mit IC's von anderen Herstellern auftraten.

Nach diesen Änderungen sollte aber wirklich alles funktionieren. Ein weiterer Test vor dem Anschluß an den Rechner ist die Messung der Stromaufnahme (5V Netzteil anschließen). Der Strom bei aktiver Karte beträgt ca. 110 mA, nach dem Ausschalten der Versorgungsspannung sollte der Strom, der vom Akku zu den Ram's fließt nur noch ca. 4 Microampere betragen.

Übrigens für diejenigen, die es noch nicht wissen: "Eine Platine ist eine kupferbeschichtete Epoxydharzplatte (gedruckte Schaltung) und nicht die funktionstüchtige Erweiterung". Die Bauteile müssen noch in die Platine eingelötet werden.

Wolfgang Enders

Und dann noch einige Tips und Tricks

1.) Die Symbole
_ Line
@ at Sighn
\ reverse Slant
[open Brcket
] close Bracket
sind vollwertig als Variablen verwendbar wie z.B.:

_=1000 oder FOR a=1 TO 3 :: READ @§ :: NEXT @

2.) Ein wirksames Codeschloß ergibt:
RUN "!!!!!!!!!!!!!! (die ganzen 5 Zeilen voll) !!!"

Dieses kann in einem Programm stehen,wo z.B. eine Passwortabfrage gemacht wird.Ist der Benutzer nich Privelegiert so sperrt sich der Rechner total beim anspringen dieser Zeile.Kein QUIT mehr möglich!!!
Es können aber auch alle anderen Buchstaben und Symbole verwendet werden.Es treten dann nur unterschiedliche ~~auf-dem~~ Effekte auf dem Bildschirm auf,oder der Soundgenerator macht sich lautstark bemerkbar (bei § z.B.)!

Tip!!!

Wen nervt es nich auch im normaler TI-Basic beim editieren immer das Wort EDIT einzugeben.Da haben es die X-Basic nutzer es besser wird manch einer denken.Aber das was die X-Basic besitzer können, können die TI-Basic benutzer schon lange.Einfach die Zeilennummer eingeben und dann "FCNT"-"X" oder"E" drücken.Aufpassen mit ENter, sonst ist die Zeile weg.Also z.B.: 100 -- FCNT X!!!
Dieses ist auch in keinem Hanbuch dokumentiert,und ist mir zufällig aufgefallen als ich im TI-Basic gewohnheitsmäßig FCNT X drückte. Und damit ist das nervige eintippen von EDIT beseitigt!

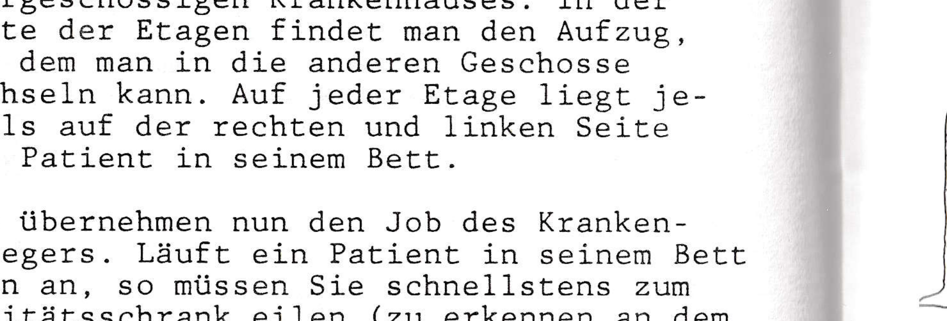
***** DAS EXTENDED-BASIC PROGRAMM *****

Aus einer großen Zahl von Einsendungen haben wir diesmal eine gut gelungene Version der berühmten Etagenspiele ausgewählt.

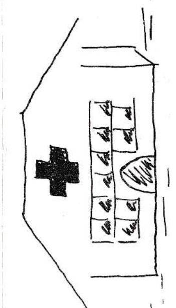
Das Spiel heißt "HOSPITAL".

Es besteht aus zwei Teilen. Im Ersten wird das Spiel erklärt, und dann wird der Hauptteil automatisch nachgeladen. Beachten Sie bei der Eingabe also die Zeilennummern.

Nach einer gelungenen Einleitung erscheint auf dem Bildschirm der Querschnitt eines viergeschossigen Krankenhauses. In der Mitte der Etagen findet man den Aufzug, mit dem man in die anderen Geschosse wechseln kann. Auf jeder Etage liegt jeweils auf der rechten und linken Seite ein Patient in seinem Bett.



Sie übernehmen nun den Job des Krankenpflegers. Läuft ein Patient in seinem Bett grün an, so müssen Sie schnellstens zum Sanitätsschrank eilen (zu erkennen an dem roten Kreuz) und eine Spritze holen. Der Pfleger wird dabei über den Joystick gesteuert. Erreichen Sie den Patienten rechtzeitig, so können Sie ihm die Spritze geben (Drucktaste) und ihn vor dem sicheren Tod bewahren.



Neben einer geschickten Wegeinteilung und rechtzeitiger Beschaffung der Spritze, müssen Sie sich vor Krankenhausgespenstern in Acht nehmen. Also ran an die Arbeit!

Damit die Sache auf Dauer nicht langweilig wird, sind natürlich verschiedene Schwierigkeitsgrade wählbar.


```
100 CALL MAGNIFY(4)
110 CALL COLOR(1,2,2):: CALL SCREEN(12)
120 CALL VCHAR(1,1,32,800)
130 CALL CHAR(100,"6B2B1C080702020605090406020
202032274D454EC44444CB4A4A4A8A8A
8A8")
140 CALL CHAR(104,"3C1E1FE7F77375564676965956 5
6554B64141458B4B2B454D4D5B9B5B5B
6E8")
150 CALL CHAR(108,"003030707070505050A0A0E0202
060C0000000000000000000000000000
000")
160 CALL CHAR(112,"000000000000000000000000001
C1EE70000000000000000000000000010
323")
170 CALL CHAR(116,"000000000000000040E0E0707070
40400000000000000000000000000808
080")
180 CALL CHAR(120,"000000002058868DF8F0F0310D0
30000000000000000804060A0334C840
403")
190 CALL CHAR(124,"000000000000002010000000000
00000000000000000008060180601000
000")
200 !
210 FOR T=1 TO 7 :: X=X+4
220 CALL SPRITE(#T,96+X,16,100,100)
230 NEXT T
240 CALL LOCATE(#6,37,105,#5,23,111,#2,68,100)

250 CALL LOCATE(#7,25,73,#4,36,106,#3,65,131)
260 CALL COLOR(#6,5,#7,5)
270 !
280 A$="  ECKI SOFT PRAESENTIERT"
290 FOR T=1 TO LEN(A$)
300 CALL HCHAR(2,2+T,ASC(SEG$(A$,T,1)))
310 CALL SOUND(-10,-7,0):: NEXT T
320 X=0 :: A$="HOSPITAL"
330 FOR T=1 TO LEN(A$)
340 CALL HCHAR(6+T,10+X,ASC(SEG$(A$,T,1)))
350 CALL SOUND(-10,-7,0):: NEXT T
360 X=0 :: DISPLAY AT(23,14):"TASTE" :: FOR T=
1 TO 100
```

```
100 CALL MAGNIFY(1):: CALL DELSPRITE(ALL):: CA
LL CHARSET
110 FOR T=0 TO 14 :: CALL COLOR(T,16,2):: NEXT
 T :: CALL SCREEN(8)
120 CALL CHARSET
130 CALL VCHAR(1,1,32,800)
140 DISPLAY AT(2,1):"loese bitte die alpha loc
k  taste"
150 DISPLAY AT(10,1):"welchen schwierigkeitsgr
ad  willst du spielen?"
160 DISPLAY AT(13,1):"druecke:"
170 DISPLAY AT(14,1):" 1  fuer schwer
     2  fuer leicht                3
  fuer sehr leicht"
180 LEB=5 :: TOT=0 :: SPR=1 :: PU=49
190 FOR T=65 TO 90
200 CALL CHARPAT(T,B$):: CALL CHAR(T+32,B$)
210 NEXT T :: A$,B$="" :: X,T=0
220 CALL CHAR(41,"18243D5EBC2426101824BC763D24
6408",45,"18187E7E181818FF")
230 CALL SPRITE(#2,41,16,73,1,#3,41,16,113,1,#
4,41,16,153,1,#5,41,16,177,1)
240 RANDOMIZE
250 CALL CHAR(33,"18083C4A08181610")
260 CALL CHAR(35,"18082C1A081C1202")
270 CALL CHAR(34,"1808081C2C180C08")
280 CALL CHAR(36,"1808081C2C180C08")
290 CALL CHAR(38,"1810103834183010")
300 CALL CHAR(37,"1810345810384840")
310 CALL CHAR(39,"18103C5210186808")
320 CALL CHAR(40,"1810103834183010")
330 CALL CHAR(72,"FCFCFC00E7E7E700")
340 CALL CHAR(64,"FFE7E78181E7E7FF")
350 CALL SPRITE(#1,33,16,73,110)
360 DISPLAY AT(24,1):" by michael eckert 1985"

370 CALL KEY(0,K,S):: IF K<49 OR K>51 THEN 370

380 IF K=49 THEN ZEIT=20
390 IF K=50 THEN ZEIT=30
400 IF K=51 THEN ZEIT=40
410 K=0
```



```
370 CALL SOUND(100,INT(RND*100)+110,0,INT(RND*
50)+130,3)
380 X=X+1 :: CALL COLOR(5,X,2,6,X,2,7,X,2):: I
F X=16 THEN X=3
390 CALL KEY(0,K,S):: IF S=0 THEN 400 ELSE 410

400 NEXT T
410 CALL SCREEN(2):: CALL CLEAR :: DISPLAY AT(
2,2):"@ BY MICHAEL ECKERT 1985"
420 FOR T=3 TO 14 :: CALL COLOR(T,INT(RND*13)+
3,2):: NEXT T
430 READ A$
440 IF A$="END" THEN 520
450 FOR T=1 TO LEN(A$)
460 CALL HCHAR(18+Y,2+T,ASC(SEG$(A$,T,1)))
470 CALL SOUND(100,-7,3)
480 NEXT T
490 Y=Y+2 :: IF Y+18>24 THEN Y=0 :: CALL HCHAR
(18,1,32,230)
500 GOTO 430
510 !
520 !
530 DATA "SIE HABEN EINEN JOB","ALS KRANKENPFL
EGER ","ANGENOMMEN UND SIE ","MUES
SEN JETZT EINE GANZE"
540 DATA "STATION VERSORGEN","LAUFEN SIE ZUM S
ANITAETS-","SCHRANK UND ENTNEHMEN
SIE"
550 DATA "EINE SPRITZE(KNOPFDRUCK)","GEHEN SIE
 NUN ZUM KRANKEN","(ER LAUFT GRUEN
 AN)"
560 DATA "UND GEBEN SIE IHM DIE","SPRITZE(KNOF
DRUCK)"
570 DATA "SIE KOENNEN AUCH MIT DEM "
580 DATA "AUFZUG FAHREN INDEM SIE","ZU IHM GEH
EN UND IN DIE "
590 DATA "GEWUENSCHTE RICHTUNG ","DRUECKEN","L
ASSEN SIE SICH NICHT VON","DEN KRA
NKENHAUSGESPENSTERN","ERWISCHEN"
600 DATA "HAUPTPROGRAMM WIRD JETZT","NACHGELAD
EN","END"
610 CALL CLEAR :: RUN "CS1"
```

```
420 CALL CLEAR :: FOR T=1 TO 14 :: CALL COLOR(
T,16,16):: NEXT T
430 GOSUB 950
440 CALL COLOR(0,2,2,1,16,2,2,16,2,3,16,2,4,16
,2,5,15,7,6,10,2,7,2,13,8,15,2)
450 CALL COLOR(9,16,2,10,16,2,11,16,2,12,16,2,
14,12,2,13,12,2)
460 CALL CHAR(136,"0090BBBC9380FFC0",137,"0001
C565DD01FF030")
470 CALL CHARPAT(136,A$):: CALL CHAR(80,A$)::
CALL CHARPAT(137,A$):: CALL CHAR(8
1,A$):: A$="" :: A=1
480 CALL CHAR(128,"6666667E7E6666663C7E6666666
67E3C1E3C60381C063C783C66667C60606
060")
490 CALL CHAR(132,"3C18181818183C7E7E1818181
81818183C6666FFFFC3C36060606060607
E7E")
500 FOR T=1 TO 8
510 CALL HCHAR(1,12+T,127+T)
520 CALL SOUND(100,300,0):: CALL SOUND(100,200
,0):: CALL SOUND(120,250,0):: NEXT
 T
530 DISPLAY AT(24,4)SIZE(22):"druecke den feue
rknopf"
540 CALL COLOR(13,4,3,13,5,6,13,7,8,13,9,3,13,
14,2,13,16,5):: CALL KEY(2,X,Y)::
IF Y=0 THEN 540
550 CALL HCHAR(24,1,72,32)
560 CALL JOYST(2,X,Y)
570 CALL POSITION(#1,D,E):: B=D/8+1 :: C=E/8+1

580 IF Y<>0 THEN CALL GCHAR(B,C,F)ELSE 660
590 IF F<>62 THEN 660
600 CALL MOTION(#1,0,0):: IF D=73 AND Y=4 THEN
 660
610 IF D=177 AND Y=-4 THEN 660
620 IF D=177 AND Y=4 THEN CALL LOCATE(#1,153,E
):: GOTO 660
630 IF D=153 AND Y=-4 THEN CALL LOCATE(#1,177,
E):: GOTO 660
640 IF Y=-4 THEN CALL LOCATE(#1,D+40,E):: GOTO
 660
```

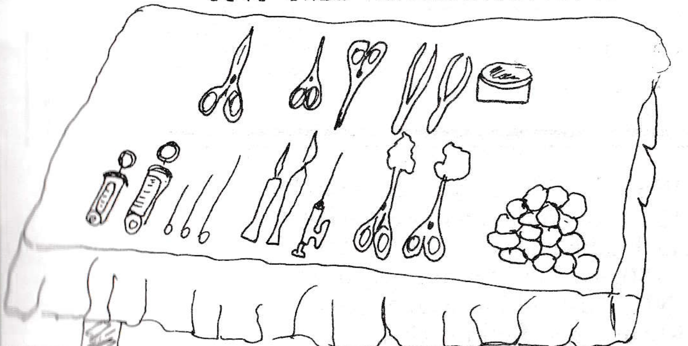


```
650 IF Y=4 THEN CALL LOCATE(#1,D-40,E)
660 IF X=-4 THEN CALL MOTION(#1,0,-5):: CALL P
ATTERN(#1,32+A):: CALL SOUND(-10,-
7,0)
670 CALL COINC(ALL,J):: IF J THEN CALL SOUND(-
1000,-7,0):: LEB=LEB-1 :: DISPLAY
AT(3,11)SIZE(1):USING "#":LEB
680 CALL PATTERN(#2,41,#3,41,#4,41,#5,41)
690 IF X=4 THEN CALL MOTION(#1,0,5):: CALL PAT
TERN(#1,36+A):: CALL SOUND(-10,-7,
0)
700 A=A+1 :: IF A>4 THEN A=1
710 IF X=0 AND Y=0 THEN CALL MOTION(#1,0,0)
720 IF LEB<1 OR TOT=7 THEN 1170
730 IF PU>50 THEN GOSUB 1300
740 S=INT(RND*3)+1 :: IF S<>2 THEN 760
750 S=INT(RND*3)+1 :: CALL MOTION(#2,0,1-S,#3,
0,3-S,#4,0,6-S,#5,0,0-S)
760 CALL PATTERN(#2,42,#3,42,#4,42,#5,42)
770 CALL COINC(ALL,J):: IF J THEN CALL SOUND(-
900,-7,0):: LEB=LEB-1 :: DISPLAY A
T(3,11)SIZE(1):USING "#":LEB
780 CALL JOYST(2,X,Y):: IF X=-4 THEN CALL MOTI
ON(#1,0,-5):: CALL PATTERN(#1,32+A
):: CALL SOUND(-10,-7,0)
790 IF X=4 THEN CALL MOTION(#1,0,5):: CALL PAT
TERN(#1,36+A):: CALL SOUND(-10,-7,
0)
800 A=A+1 :: IF A>4 THEN A=1
810 IF X=0 AND Y=0 THEN CALL MOTION(#1,0,0)
820 CALL KEY(2,X,Y):: IF Y=0 THEN 880
830 CALL GCHAR(B,C,F)
840 IF F=32 THEN 880
850 IF SPR=1 AND F=81 THEN PU=PU+7 :: DISPLAY
AT(4,24)SIZE(2):USING "##":PU
860 IF SPR=1 AND F=81 THEN SPR=0 :: KR,KRA=0 :
: CALL HCHAR(B,C-1,136):: CALL HCH
AR(B,C,137):: DISPLAY AT(4,12)SIZE(4):"nein" :
: GOTO 930
870 IF F=64 THEN SPR=1 :: DISPLAY AT(4,12)SIZE
(4):"ja  "
880 IF KR=1 THEN KRA=KRA+1 :: CALL GCHAR(K1(L)
,K2(L),F):: IF F=80 AND KRA>ZEIT T
HEN GOSUB 1220
890 IF KR=1 THEN 930
```

```
900 KR=1 :: L=L+1 :: CALL GCHAR(K1(L),K2(L),F)
:: IF F<>136 THEN GOTO 900
910 IF L>7 THEN L=0
920 CALL HCHAR(K1(L),K2(L),80):: CALL HCHAR(K1
(L),K2(L)+1,81)
930 !
940 GOTO 560
950 CALL HCHAR(24,1,72,32):: CALL VCHAR(1,1,72
,48):: CALL VCHAR(1,31,72,48):: CA
LL HCHAR(1,1,32,160):: CALL HCHAR(1,8,88,18)
960 FOR T=3 TO 7 :: X=X-1 :: CALL HCHAR(6+X,T,
89):: NEXT T
970 X=0 :: FOR T=4 TO 7 :: X=X-1 :: CALL HCHAR
(6+X,T,90):: NEXT T
980 X=0 :: FOR T=30 TO 26 STEP -1 :: X=X-1 ::
CALL HCHAR(6+X,T,91):: NEXT T
990 X=0 :: FOR T=29 TO 26 STEP -1 :: X=X-1 ::
CALL HCHAR(6+X,T,92):: NEXT T
1000 FOR T=1 TO 7 :: READ A :: K1(T)=A :: NEXT
 T :: FOR T=1 TO 7 :: READ A :: K2
(T)=A :: NEXT T
1010 FOR T=1 TO 7 :: CALL HCHAR(K1(T),K2(T),13
6):: CALL HCHAR(K1(T),K2(T)+1,137)
:: NEXT T
1020 CALL HCHAR(6,1,72,32)
1030 CALL HCHAR(21,3,88,28):: CALL HCHAR(16,3,
88,28):: CALL HCHAR(11,3,88,28)
1040 CALL CHAR(88,"FF5555FFFF5555FF",89,"01030
40D13374FDE",90,"3D7AF4E8D0A04080"
)
1050 CALL CHAR(91,"80C020B0C8ECF27B",92,"BC5E2
F170B050201")
1060 CALL CHAR(59,"4AAAEAAAA4000000A6A8AAAA460
00000EE82C4888E00",62,"FF8199A5998
18181")
1070 FOR T=1 TO 8 :: READ A,B :: CALL HCHAR(A,
B,59):: CALL HCHAR(A,B+2,60):: CAL
L HCHAR(A,B+1,61):: CALL HCHAR(A+1,B+1,62):: N
EXT T
```



```
1080 CALL HCHAR(23,3,64,3):: CALL HCHAR(22,3,6
4,3)
1090 DISPLAY AT(3,5)SIZE(8):"leben:5"
1100 DISPLAY AT(4,4)SIZE(12):"spritze:ja  "
1110 DISPLAY AT(3,18)SIZE(7):"tote:0"
1120 DISPLAY AT(4,17)SIZE(8):"punkte:0"
1130 RETURN
1140 DATA 23,15,20,10,15,20,10
1150 DATA 28,28,4,28,4,28,4
1160 DATA 9,8,9,21,14,8,14,21,19,8,19,21,22,8,
22,21
1170 CALL DELSPRITE(ALL)
1180 DISPLAY AT(13,5):"noch ein mal? ja/nein "
 :: ACCEPT AT(18,13)BEEP SIZE(4):A
$
1190 IF A$="ja" THEN RUN
1200 IF A$="nein" THEN PRINT : : :"DANN EBEN N
ICHT" :: END
1210 GOTO 1170
1220 !
1230 CALL MOTION(#1,0,0,#2,0,0,#3,0,0,#4,0,0,#
5,0,0)
1240 FOR T=1 TO 190 STEP 3 :: CALL SOUND(-1000
,300-T,0):: NEXT T
1250 CALL SOUND(-700,-7,0)
1260 DISPLAY AT(K1(L),K2(L)-2)SIZE(2):"- "
1270 KR,KRA=0 :: TOT=TOT+1 :: DISPLAY AT(3,23)
SIZE(1):USING "#":TOT :: PU=PU-5
1280 IF PU<0 THEN PU=0
1290 RETURN
1300 DISPLAY AT(24,13)SIZE(5):"bonus"
1310 CALL SOUND(-1000,250,0)
1320 CALL SOUND(-2000,270,0)
1330 PU=0 :: LEB=LEB+1
1340 CALL HCHAR(24,1,72,28):: RETURN
```



ASSEMBLERKURS II

*** ASSEMBLERKURS TEIL II ***

Am Ende des ersten Teils ist ein Programm zum Löschen des Bildschirms besprochen worden. Er wird dazu komplett mit Leerzeichen beschrieben. Um nun den darstellbaren Zeichensatz des TI's aufzulisten ist nur eine kleine Änderung notwendig. Damit die Zeichen auf der Mitte des Bildschirms erscheinen, laden wir in Register 0 als Anfangsadresse die Zahl 320. In jeder Zeile werden 32 Zeichen dargestellt. Also muß nun am Anfang der 10. Zeile das erste Zeichen erscheinen. Nachdem ein Buchstabe in den VDP-Ram geschrieben worden ist, nehmen wir das nächsten ASCII-Zeichen. Dazu müssen wir den Wert in Register 0 um 1 erhöhen. Da der ASCII-Code des Buchstabens im ersten, dem höherwertigen Byte von Register 1 steht, müssen wir jeweils >0100 auf den alten Wert addieren. Dazu gibt es den Befehl AI (Add immediate). Er hat die Form AI x,y , wobei x die Angabe eines Registers ist, und y die zu addierende Zahl. Er bedeutet also: addiere (add) auf den Inhalt des angegebenen Registers die direkt nachfolgende Zahl (immediate).

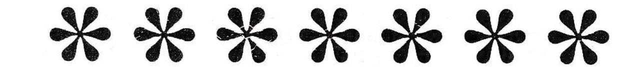
AI 1,>0100

Jetzt fehlt noch die Abfrage, wie oft das Programm ein Zeichen in den VDP-Ram schreiben soll. Wie im ersten Teil des Kurses beschrieben, wollen wir die ASCII-Zeichen von 32 bis 128 aufgelistet haben. Also fragen wir ab, ob das Zeichen 128 erreicht worden ist. In Register 1 muß dann der Wert >E000 stehen (128 plus dem Offset von 96 ergibt den Wert von 224, hexadezimal >E0. Wir vergleichen nun den Inhalt von Register 1 mit der Zahl >E000 über den Befehl CI (Compare immediate), der genauso wie der Befehl AI adressiert wird.

CI 1,>E000

Wie beim ersten Programmteil soll das Programm in einer Schleife zum Befehl für die Ausgabe des Zeichens mit dem Befehl JNE LOOP2 zurückspringen, falls der Wert von >E000 noch nicht erreicht worden ist (JNE= Jump if not equal - springe, falls nicht gleich). Das komplette Programm sieht dann wie folgt aus:

```
 MYWS BSS 32       reserviert Speicherplatz für eigene Register
START LWPI MYWS    lädt die eigenen Register
      LI 0,0       1. Zeichen auf dem Bildschirm
      LI 1,>8000   dargestellte Zeichen soll Leerzeichen sein
      LI 2,>300    letztes Zeichen auf dem Bildschirm
LOOP1 BLWP @>2020  schreibt Zeichen
      INC 0        erhöhe die Adresse um 1
      C 0,2        vergleiche die Adresse mit oberer Grenze
      JNE LOOP1    falls nicht erreicht, Wiederholung
      LI 0,320     lädt als Anfangsadresse die 10. Zeile
LOOP2 BLWP @>2020  schreibt das Zeichen
      INC 0        erhöhe die Bildschirmadresse um 1
      AI 0,>0100   nimm das nächste ASCII-Zeichen
      CI 0,>E000   ist das Ende erreicht ?
      JNE LOOP2    falls nicht, Wiederholung
      LWPI >83E0   lade die Register vom Basic
      RT           kehre zum Basic zurück
      END
```



In diesem kurzen Programm sind für das gleiche Problem zwei verschiedene Lösungswege gewählt worden. Um bei der Schleife zur Ausgabe der Zeichen auf den Bildschirm die obere Grenze abzufragen, wurde einmal der Befehl C und dann CI benutzt. Bei C (Compare) wurden die Inhalte von zwei Registern miteinander verglichen. Bei CI dagegen der Inhalt eines Registers direkt mit der nachfolgenden Zahl verglichen.
Es gibt also verschiedene Methoden der Adressierung, die jeweils von dem benutzten Befehl abhängen. Man unterscheidet zwischen folgenden Adressierungsarten:

A - Allgemeine Adresse

Hier gibt es mehrere Varianten. Einmal kann man direkt ein Register angeben. Der Befehl arbeitet dann mit dem Wert, der in diesem Register steht. Ebenso ist eine indirekte Adressierung möglich. Sie wird durch ein * vor der Angabe des Registers gekennzeichnet. Der Befehl arbeitet dann mit dem Wert, der auf der Adresse steht, die in dem Register angegeben wird.

MOV 4,8 bedeutet also: kopiere den Inhalt von Register 4 in das Register 8; MOV *4,*8 dagegen, kopiere den Inhalt der Adresse, die in Register 4 steht auf die Adresse, die in Register 8 angegeben wird. Wenn es auch am Anfang etwas kompliziert erscheint, so werden im Laufe des Kurses die Vorteile dieser Adressierungsart klar werden.

Anstelle der Register kann man auch eine Speicheradresse angeben. Dabei muß vor die Adresse ein @ gestellt werden. MOV @>1000,@LABEL heißt: Speichere den Inhalt der Adresse >1000 auf die Adresse, die durch das Label LABEL angegeben wird.

Dann gibt es noch die indirekte Speicheradressierung. MOV 7,@LABEL(5) bedeutet: Speichere den Inhalt von Register 7 auf die Adresse, die durch die Summe von LABEL und den Inhalt von Register 5 errechnet wird. Diese Methode ist aber schon etwas für den fortgeschrittenen Programmierer.

Als letzte Möglichkeit gibt es die indirekte Registeradressierung mit automatischer Erhöhung des Registerinhalts. C *3+,2 bedeutet: Vergleiche den Inhalt von jener Adresse, die in Register 3 steht, mit dem Inhalt von Register 2 und erhöhe die Adresse in Register 3 um den Wert 2 (Eine Wortlänge besteht aus 2 Bytes; bei Byteoperationen wird die Adresse demnach nur um den Wert 1 erhöht). Alle hier aufgeführten Befehle lassen sich beliebig miteinander kombinieren.

W - Workspace Adressierung

Hier ist nur die direkte Angabe eines Registers erlaubt.

PC - Programmzähler Adressierung

In diesem Fall wird eine relative Adresse zum Programmzähler erwartet. Der Programmzähler gibt dem Microprozessor an, an welcher Speicherstelle der Befehl steht, der gerade bearbeitet wird.

JMP LOOP bedeutet also, daß an der Adresse, die durch das Label LOOP angegeben wird, der nächste auszuführende Befehl steht (JMP= Jump, springe). Das Assemblerprogramm berechnet den Abstand der Adresse von LOOP und der Adresse des Programmzählers und speichert die Differenz im Maschinencode ab. Da dafür bei dieser Adressierungsart nur ein Byte zur Verfügung steht, darf die Differenz der Adressen maximal +/- 128 Worte betragen. Ansonsten wird nicht korrekt assembliert. Dies ist bei langen Programmen unbedingt zu beachten.

I - Direkte (immediate) Adressierung

Hier ist die direkte Angabe eines Wertes gefordert, z.B. LI 0,>0100: lade Register 0 mit dem Wert >0100. Der zweite Operand ist also durch direkte Adressierung angegeben worden.

CRU - CRU Bit Adressierung

Diese Adressierungsart wird erst in weiterer Zukunft bei der Behandlung der CRU-Befehle erklärt werden.

Auf der nächsten Seite folgt nun eine Liste aller Assemblerbefehle mit Angabe der Adressierungsarten, dem Basismaschinencode, in den der Befehl vom Assembler umgewandelt wird, das Format, in der die Adressinformation verarbeitet wird und die Status-Bits, die vom Befehl verändert werden können.

Die Angabe des Adressformats ist für den Anfang unwichtig. Sie wird erst bei intensiver Beschäftigung mit dem Assembler interessant und daher später erklärt. Der Vollständigkeit wegen werden in der Liste jedoch jetzt schon der Maschinencode und die Formatangabe aufgelistet.

Wichtig dagegen ist die Beeinflussung des Statusbytes von den jeweiligen Assemblerbefehlen. Dazu ausführlicher:

Der Microprozessor besitzt drei interne Register. In Ihnen steht 1. die Adresse, an der im Speicher der Platz für die Register reserviert ist. Er muß ja bei Angabe eines Registers wissen, wo der zu bearbeitende Wert im Speicher steht.
In diesem Punkt unterscheidet sich der TMS 9900 ganz entscheident von anderen Prozessoren. Seine Arbeitsregister sind nicht hardwaremäßig im Chip realisiert, sondern stehen im Speicher. Wie Sie sich erinnern, haben wir immer am Anfang des Programms mit LWPI die Startadresse unseres Workspace dem Prozessor "mitgeteilt". Das 2. interne Register enthält den Programmzähler, der schon vorher beschrieben worden ist. Das 3. interne Register wird das Statusregister genannt. In ihm wird angezeigt, ob bei einer Rechenoperation eine Null herausgekommen ist, ein negatives Ergebnis oder ein Positives. Ebenso, ob bei einem Vergleich von zwei Werten beide identisch waren oder nicht. Dazu werden jeweils in diesem Register Bits gesetzt, was im Anschluß an diese Liste näher erklärt wird.

Legen Sie sich diese Befehlsübersicht zur Seite, und Sie haben wie bei der Referenzkarte zum Basic eine schnelle Übersicht aller Assemblerbefehle und deren Funktion, was beim Programmieren als Anfänger bestimmt sehr nützlich ist.



B - Kurzbezeichnung (Mnemonic) der Assemblerbefehle
FO - Erster Operand
SO - Zweiter Operand
MC - Maschinencode
F - Format
SB - Statusbits, die vom Befehl verändert werden können

Befehl	B	FO	SO	SB	MC	F
Addiere Wörter	A	A	A*	0-4	A000	1
Addiere Bytes	AB	A	A*	0-5	B000	1
Bilde Absolutwert	ABS	A	-	0-4	0740	6
Addiere direkt	AI	WR*	I	0-4	0220	8
Logisch UND direkt	ANDI	WR*	I	0-2	0240	6
Verzweige	B	A	-	---	0440	6
Verzweige und verbinde	BL	A	-	---	0680	6
Verzweige und lade WP	BLWP	A	-	---	0400	6
Vergleiche Wörter	C	A	A	0-2	8000	1
Vergleiche Bytes	CB	A	A	0-2,5	9000	1
Vergleiche direkt	CI	WR	I	0-2	0280	8
Lösche	CLR	A	-	---	04C0	6
Vergleiche Bits mit 1	COC	A	WR	2	2000	3
Vergleiche Bits mit 0	CZC	A	WR	2	2400	3
Verringere um 1	DEC	A	-	0-4	0600	6
Verringere um 2	DECT	A	-	0-4	0640	6
Dividiere	DIV	A	WR*	4	3C00	9
Erhöhe um 1	INC	A	-	0-4	0580	6
Erhöhe um 2	INCT	A	-	0-4	05C0	6
Invertiere	INV	A	-	0-2	0540	6
Springe bei "gleich"	JEQ	PC	-	---	1300	2
Springe bei "größer"	JGT	PC	-	---	1500	2
Springe bei "mehr"	JH	PC	-	---	1B00	2
Springe "mehr,gleich"	JHE	PC	-	---	1400	2
Springe "weniger"	JL	PC	-	---	1A00	2
Springe "weniger,gleich"	JLE	PC	-	---	1200	2
Springe bei "kleiner"	JLT	PC	-	---	1100	2
Springe ohne Bedingung	JMP	PC	-	---	1000	2
Springe "Übertrag=0"	JNC	PC	-	---	1700	2
Springe "nicht gleich"	JNE	PC	-	---	1600	2
Springe "Überlauf=0"	JNO	PC	-	---	1900	2
Springe "Übertrag=1"	JOC	PC	-	---	1800	2
Springe "Parität=1"	JOP	PC	-	---	1C00	2
Lade CRU	LDCR	A	Note 1	0-2,5	3000	4
Lade direkt	LI	WR*	I	0-2	0200	8
Lade Interrupt Maske	LIMI	I	-	12-15	0300	8
Lade Workspace Adresse	LWPI	I	-	---	02E0	8
Übertrage Wort	MOV	A	A*	0-2	C000	1
Übertrage Byte	MOVB	A	A*	0-2,5	D000	1
Multipliziere	MPY	A	WR*	---	3800	9
Bilde den negativen Wert	NEG	A	-	0-4	0500	6
Oder-Verbindung direkt	ORI	WR*	I	0-2	0260	8
Rücksprung mit neuem WP	RTWP	-	-	0-6,12-15	0380	7
Subtrahiere Wort	S	A	A*	0-4	6000	1
Subtrahiere Byte	SB	A	A*	0-5	7000	1
Setze CRU-Bit auf 1	SBO	CRU	-	---	1D00	2
Setze CRU-Bit auf 0	SBZ	CRU	-	---	1E00	2
Setze alle Bits auf 1	SETO	A	-	---	0700	6
Schiebe nach links	SLA	WR*	Note 2	0-4	0A00	5
Oder-Verknüpfung Wort	SOC	A	A*	0-2	E000	1
Oder-Verknüpfung Byte	SOCB	A	A*	0-2,5	F000	1

Befehl	B	FO	SO	SB	MC	F
Schiebe nach rechts	SRA	WR*	Note 2	0-3	0800	5
Schiebe rechts im Kreis	SRC	WR*	Note 2	0-3	0B00	5
Schiebe rechts logisch	SRL	WR*	Note 2	0-3	0900	5
Speichere CRU	STCR	A*	Note 1	0-2,5	3400	4
Speichere Statusregister	STST	WR	-	---	02C0	8
Speichere den Workspace	STWP	WR	-	---	02A0	8
Tausche beide Bytes um	SWPB	A	-	---	06C0	6
UND-Verknüpfung pro Wort	SZC	A	A*	0-2	4000	1
UND-Verknüpfung pro Byte	SZCB	A	A*	0-2,5	5000	1
Teste CRU-Bit	TB	CRU	-	2	1F00	2
Führe Befehl aus	X	A	-	---	0480	6
Extended Operation	XOP	A	Note 3	6	2C00	9
Exklusiv-OR Verbindung	XOR	A	WR*	0-2	2800	3

Note 1: Der zweite Operand gibt die Anzahl der Bits an, die übertragen werden sollen.

Note 2: Der zweite Operand definiert die Anzahl der Bits, um die der Inhalt des Registers verschoben werden soll. Der Wert muß zwischen 0 und 15 liegen. Ist er =0 dann wird die Zahl genommen, die in Register 0 steht.

Note 3: XOP erlaubt softwaremäßig eine Erweiterung des Befehlssatzes. Mit diesem Befehl wird zu einem Unterprogramm verzweigt. Diese Funktion gibt es nur auf dem TI99/4A. Dabei gibt es zwei Versionen. Welche davon beherrscht Ihr TI ? Peeken Sie dazu das Wort an der Adresse >44. Steht dort °FFE8 kann Ihr TI nur XOP 2, bei >FFD8 XOP 1 und XOP 2.

Die einzelnen Bits des Statusregisters haben folgende Bedeutung:

```
Bit 0 - LGT - Logisch größer als
  1   - AGT - Arithmetisch größer als
  2   - EQ  - gleich
  3   - C   - Übertrag
  4   - OV  - Überlauf
  5   - OP  - Parität (wenn Summe der 1er ungerade ist)
  6   - X   - Extended Operation wird gerade ausgeführt
 7-11 -   wird nicht benutzt
12-15 - IM  - Interrupt Maske (Zahl der erkennbaren Interrupts)
```

Beispiel: C 2,3 - Vergleiche den Inhalt der Register 2 und 3. Steht in beiden Registern der gleiche Wert, so wird das Bit 2 (Equal) auf 1 gesetzt. Ist dies nicht der Fall, so steht hier eine 0. Ist die Zahl in Register 2 größer als die in Register 3, so wird Bit 0 gesetzt. Dabei muß man beachten: Man unterscheidet zwischen logisch und arithmetisch größer. Bei logisch betrachtet man den Registerinhalt als immer positive hexadezimale Zahl. Bei arithmetisch werden alle Zahlen, die größer oder gleich >8000 sind als negative Zahlen (Zweier-Komplement) aufgefaßt. Also >9345 ist logisch größer als >0110, aber arithmetisch kleiner !





Steht in der Tabelle ein * hinter der Angabe zur Adressierung, so wird in der angegebenen Adresse das Ergebnis des Assemblerbefehls abgespeichert.

Beispiel: A 2,3

Addiere Register 2 mit Register 3. In diesem Fall wird das Ergebnis der Rechenoperation in das Register 3 geschrieben. Der vorher dort abgespeicherte Summand wird vom Ergebnis überschrieben. Bei den Befehlen MPY (Multiplizieren) und DIV (Dividiere) ist als Ausnahme zu beachten, daß das Ergebnis in dem Zielregister und dem darauf folgenden Register abgespeichert wird. Bei diesen Rechenoperationen kann ja beim Ergebnis die doppelte Stellenanzahl zur Darstellung notwendig sein.

Nach soviel Theorie gehen wir nun wieder zur Praxis über. Die letzten drei Seiten dienen als Nachschlagwerk, müssen also nicht unbedingt verstanden sein. Es werden einzelne Punkte noch ausführlich behandelt und durch Beispiele vertieft werden.

Man wird auf Assemblerprogramme zurückgreifen, wenn das gleiche Problem in der Programmiersprache Basic nicht mit einer zufriedenstellenden Geschwindigkeit gelöst werden kann. Ebenso, um Funktionen des Rechners zu nutzen, die durch Basic nicht realisierbar sind. Das wichtigste dabei ist die Kommunikation zwischen Mensch und Rechner, die Eingabe über Tastatur oder Joysticks. Reaktionsspiele sind in Basic nur deswegen schwer zu programmieren, weil die Steuerung und Darstellung auf dem Bildschirm in Basic zu langsam ist. Zuerst wird daher als Weiterführung unserer ersten Programme die Ausgabe von beliebigem Text erklärt, und dann die Abfrage der Tastatur.

Bisher können wir den Bildschirm löschen, und einzelne Zeichen auf den Bildschirm bringen. Soll nun ein bestimmter Text dargestellt werden, so können wir ihn mit dem Befehl TEXT in den Speicher schreiben. Zu Beginn eines Assemblerprogramms reserviert man Speicherplatz für die Register, vom Programm benötigte Puffer und definiert im Programm fest benutzte Variabeln.

```
AUSGABE TEXT 'DIES IST EIN TEST'
```

schreibt in den Speicher von Beginn der Adresse Ausgabe an den ASCII-Code der Zeichen, die auf den Befehl TEXT in den '-Strichen folgen. Sie können hier beliebige Zeichen, also auch Zahlen angeben. Da ein neuer Befehl auf einer Speicherstelle mit gerader Adresse anfangen muß, wird bei einer ungeraden Anzahl von Buchstaben der Text mit einem Leerzeichen aufgefüllt. Als Erinnerung: In ein Wort (16-Bit) passen zwei Bytes, also zwei Zeichen. Nun geben wir an, in welcher Zeile und welcher Spalte der Text erscheinen soll. In Basic würde das z.B. ZEILE=10 lauten. In Assembler schreibt man

```
ZEILE DATA 10
SPALTE DATA 1
```

oder in hexadezimalen Zahlen >00A0 und >0001. Bei den meisten Assemblern darf man die führenden Nullen auch weglassen.

Um die genaue Bildschirmposition zu berechnen, multiplizieren wir die Zeilenzahl mit 32 (Zeichen pro Zeile). Dann wird die Spaltenzahl auf das Ergebnis addiert.

```
LI  0,32
MPY ȻZEILE,0
MOV 1,0
A   ȻSPALTE,0
```

Register 0 wird mit der Zahl 32 geladen. Dann wird der Inhalt der Adresse Zeile (dort haben wir am Anfang des Programms den Zeilenwert abgespeichert) mit dem Inhalt von Register 0 multipliziert. Das ergebnis der Multiplikation wird dabei in Register 0 und 1 abgespeichert (siehe obige Erklärung). Das Ergebnis steht in diesen 32-Bits rechtsbündig, also in unserem Fall in Register 1 (Ergebnis kleiner als 32*24=768). Da wir später die Bildschirmposition in Register 0 beim Aufruf des Hilfsprogramms gebrauchen, speichern wir Register 1 in Register 0 um. Die Spaltenzahl wird dann auf den Inhalt von Register 0 aufaddiert. Man beachte, daß bei der Addition das Register 0 als zweiter Operand angegeben werden muß. Damit erreicht man, daß das Ergebnis der Addition wieder in Register 0 abgespeichert wird.

In Register 2 wird nun die Startadresse unseres Textpuffers geladen: LI 2,AUSGABE . Jetzt folgt eine etwas trickreiche Programmierung. Mit LOOP MOV *2+,1 wird der Inhalt der Adresse, die in Register 2 steht in das Register 1 übertragen; also die ersten beiden Buchstaben. Dabei wird der Inhalt von Register 2 um den Wert 2 erhöht, die Adresse des Textpuffers. Springt man später wieder auf diesen Befehl zurück, so werden die nächsten beiden Buchstaben genommen. Da bei Basic auf den ASCII-Code der Zeichen noch ein Offset von 96 (>60) addiert werden muß, addieren wir auf das Register 1 die Zahl >6060: AI 1,>6060 . Mit BLWP Ȼ>2020 wird wie gehabt das erste Zeichen von Register 1 auf die Bildschirmposition geschrieben, die in Register 0 steht.
Mit SWPB 1 tauschen wir die Bytes von Register 1 untereinander aus, um auch den zweiten Buchstaben auf den Bildschirm schirm durch ein nochmaliges BLWP Ȼ>2020 auszugeben.

Danach fragen wir mit CI 2,ZEILE ab, ob unser ganzer Text schon in den Video-Ram übertragen worden ist. Wir vergleichen die aktuelle Textpufferadresse mit der Adresse, auf der wir die nächste Information nach dem Textpuffer abgespeichert haben. In unserem Programm demnach die Adresse ZEILE. Ist diese Adresse nicht erreicht, so soll das Programm zur weiteren Textausgabe zurückspringen, ansonsten kann das Programm beendet werden (JL LOOP). Das komplette Programm sieht dann folgendermaßen aus:

```
MYWS      BSS   32
AUSGABE   TEXT  'BELIEBIGER TEXT!'
ZEILE     DATA  10
SPALTE    DATA  1
START     LWPI  MYWS
          LI    0,32
          MPY   ȻZEILE,0
          MOV   1,0
```




```
          A     ȻSPALTE,0
          LI    2,AUSGABE
LOOP      MOV   *2+,1
          AI    1,>6060
          BLWP  Ȼ>2020
          SWPB  1
          BLWP  Ȼ>2020
          CI    2,ZEILE
          JLT   LOOP
          LWPI  >83E0
          RT
          END
```

Hinweis: Das Zeichen Ȼ wird aufgrund des deutschen Zeichensatzes vom Drucker falsch wiedergegeben. Gemeint ist das Zeichen auf der TI-Tastatur über der 2 (Klammeraffe oder AT genannt).

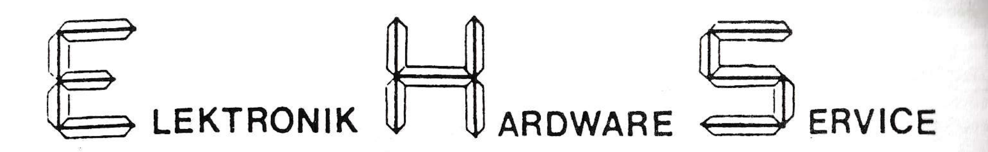
Dieses Programm erlaubt Ihnen nun an gewünschter Stelle einen beliebigen Text auf den Bildschirm zu bringen. Es ersetzt also den DISPLAY-Befehl des Extended Basic und erlaubt, mit anderen Funktionen in ein Assemblerprogramm eingebaut, eine superschnelle Textausgabe.

Noch ein Tip, falls Sie mit dem Minimem programmieren. Anfang und Ende des Programms müssen dann anders aussehen, wenn das Programm fehlerfrei laufen soll. Da das Betriebssystem gegenüber Extended-Basic anders funktioniert, müssen Sie anstatt RT am Ende des Programms B Ȼ>0070 eingeben. Oder Sie laden keine eigenen Register, lassen also MYWS BSS 32 und LWPI MYWS am Programmanfang weg, und beenden das Programm wie gewohnt mit RT bzw. B *11, was das gleiche ist. Der Grund liegt darin, daß beim Aufruf CALL LINK mit dem Minimem schon automatisch ein neuer Workspace geladen wird. Er liegt dann auf der Adresse >70B8. Dieser Kurs bezieht sich in erster Linie auf die Assemblerpakete mit Extended Basic.

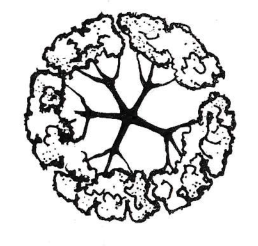
Neu auf dem Markt ist ein solches Assemblerpaket incl. Speichererweiterung von der Firma Laloire Hardware Service zum Preis von 275,- DM. Die Speichererweiterung wird seitlich an den TI angeschlossen, hat einen durchgeführten Bus und ist batteriegepuffert. Eine Kontrollampe überwacht den korrekten Anschluß. Die Erweiterung ist nur 7 cm breit und paßt sich in Tiefe und Höhe, dem TI-Gehäuse an. Dadurch ist eine stabile Verbindung gegeben, wodurch ohne Probleme weitere Geräte und auch eine Peripheriebox angeschlossen werden können. Bereits in einer alten Ausgabe des TI wurden die Vorteile einer batteriegepufferten Speichererweiterung aufgeführt. So bleiben Assembler- wie Basicprogramme nach dem Ausschalten des Rechners erhalten, wodurch die langen Ladezeiten des Cassettenrecorders umgangen werden können. Das Assemblerpaket erlaubt einen 2-Wegassembler ähnlich dem Editor/Assembler. Mit dem Nachteil der geringeren Geschwindigkeit bietet es jedoch dem User mit schmalem Geldbeutel auch ohne teurem Diskettenlaufwerk alle Möglichkeiten der Assemblerprogrammierung.

Weitere Assemblersoftware zur Sicherung von Basic-Programmen, Befehlssatzerweiterungen des Extended-Basics, hochauflösende Grafik, etc. sind auf Cassette erhältlich.

Bei Fragen zum Assemblerkurs, Verbesserungsvorschlägen oder wenn Sie eigene Assemblerprogramme geschrieben haben, dann schreiben Sie an:
Gregor Lohmann, Bleiberger Str.54, 5100 Aachen .



BÖRSE



Verkaufe Literatur: TI 99 Intern von Heiner Martin 20,- DM, TMS 9900 Microprozessor Handbuch 30,- DM bei Hans Knabel, Münsterstr.24, 7640 Kehl-Neumühl, Tel.: 07851/1482

Verkaufe 32K-Speichererweiterung, extern, batteriegepuffert und mit durchgeführtem Bus für 235,- DM. Dazu Assemblersoftware für 20,- bis 30,- DM pro Cassette.
Suche Sprachsynthesizer bis 80,- DM. Gregor Lohmann, Bleiberger Str.54, 5100 Aachen, Tel.: 0241/86447

Wer ist an folgenden Modulen(alle neu und noch nicht gebraucht) interessiert? Othello 30,-- DM, Minus Mission 20,-- DM, Alien Addition 20,-- DM sowie 1 Joystick (direkt anschlußfertig) mit 3 Feuerknöpfen Pilotengriff und 4 Saugnäpfen NEU 50,-- DM
Rainer Gawrikow, Sommerhausen 15, 5203 Much, Tel.: 02245/3983

Suche gutes Vokabelprogramm, evtl. mit englischem Grundwortschatz, gegen Bezahlung oder Tausch, sowie eine Centronics- Schnittstelle bis 150,-- DM extern!
P. Höppler, Rheinstr. 16, 8000 München 40

Verkaufe Konsole, Ex- Basic Modul, dtsch. Handbuch, Joystick, Datenrecorder mit Kabel, Modul Daten u. Analyse, externes RS232, Drucker Epson 22, viele Programme auf Cassette, sowie diverse Literatur.
Preis VB 1300,-- DM, evt. auch einzeln,
Persönliche Anlieferung bis 100 km Umkreis
Erwin Wiegel, Richtstr. 6, 8500 Nürnberg 20

Verkaufe MINI-ASSEMBLER (Firma Radix). Gibt die Befehle INIT, LOAD und LINK frei! Mit durchgeschliffenem Bus zum seitlichen Anstecken an die Konsole. Inclusive Beschreibung und Assemblersoftware für 90,- DM.
Gregor Lohmann, Bleiberger Str.54, 5100 Aachen, Tel.: 0241/86447

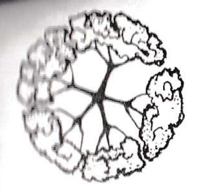
TI99/4A + Extended Basic + Schach + Kass - Rec - Kabel + Minus - Mis + Sofware + Lit; VB 520,-- DM
Norbert Flache, Tel.: 0201/306748 ab 17.30 Uhr

Verkaufe TI99/4A, orig. TI- Box mit Laufwerk, Controller, 32K- Ram, Sharp Datenrecorder LE- 152 mit Netzteil u. Recorderkabel, 1 Paar TI- Joysticks, X- Basic Modul mit dt. Handbuch, E/A- Modul + dt. Handbuch;
Module: Datenverwaltung u. Analyse, Hallenfußball, TI- Invaders, Munchman;
Diskettenprogramme 3- D World (Hi- Res Grafik in Maschine);
Diskettenunterprogramme Call Link ("Small") Kleinbuchstaben u. Umlaute, Call Link ("Scroll", X, A$) Scrollen des Schirms in 4 Richtungen, Call Link ("Spiral") Bildausblendung;
X- Basicprogramm für Poke V u. Peck V (Direktzugriff auf VD8 Ram), Super- Adressprogramm in X- Basic für 200 Adressen, mit Druckerunterstützung, Suchen nach Teilstrings, Suchen mit Platzhaltern, Suchen nach mehreren Kriterien, Name- Hashing, Sortieren nach verschiedenen Kriterien Katalogfunktion, Zugriff nur mit Code, außerdem noch über 25 Spiele in Basic und X- Basic
Angebote bitte schriftlich an: Harald Müller, Ferdinandstr. 37, 5000 Köln 80

Verkaufe 8 Bit Ausgabegerät aus der TI-Revue, Anschluß an den Soundausgang, Schaltung im Gehäuse incl. Trafo, LED- Kasten, Experimentierboard, mit Software (TI oder X-Basic) für 50,-- DM; 8 Bit Relaisplatine (einfach an Ausgabemodul ansteckbar) für 20,-- DM
Gawrikow, Haus Nr. 15, 5203 Much- Sommerhaus

Verkaufe 2 TI- Module (Statistik- u. Datenverwaltung Analyse- Modul) einzeln für je 55,-- DM, zusammen für 100DM
Klaus- Dieter Lutz, Martinstr. 13, 8707 Veitshöchheim, Tel.: 0931/95298

Verkaufe 5 Cassetten mit je 13 - 30 Pogrammen pro Cassette 8,-- DM; 1 Selbstbau- Joystickpaar (Drucktasten statt Hebel) für 12,-- DM; TI- Revue 2/3/7 1984 je 2,-- DM;
Guido Pahlberg: TI99/4A 20 Programme f. Grundversion iwt- Verlag, 217 Seiten für 8,-- DM;
Renko Hal: Superspiele 40 prog., davon 6 Ex- Bas 115 Seiten für 8,-- DM; Chip: TI99/4A 23 prog. davon 16 Ex- Basic (NP 18,-- DM) für nur 8,-- DM ferner eine A4- Mappe + Karteikarten (ca. 20) mit Utilities, Befehlsdef. (für den 1. Kunden gratis!!)
Bei Abnahme aller Teile gilt ein Sonderpreis von 65,-- DM!
Sonst gelten die Preise alle zuzügl. Porto, NN etc.
Michael Justiu, Hans- Sachs- Str. 10, 5000 Köln 41



Verkaufe folgende Software: Alpiner (M) 40,-- DM, Buck Rogers (M) 40,-- DM, Finanzberater (C) 10,-- DM, Marketing- Planspiel (C) 5,-- DM, Munch Man (M) 35,-- DM, Speech Editor (M) 40,-- DM, Statistik (M) 40,-- DM, Ti- Invaders (M) 30,-- DM, Tomstone City (M) 30,-- DM, Buchungsjournal (M) 190,-- DM, EX- Basic Compiler (D) 150,-- DM, Forth (D) 60,-- DM, Lagerverwaltung (D) 50,-- DM, Parsec (M) 40,-- DM, Personal Financial Aids (C) 10,-- DM, Schachmeister (M) 70,-- DM, The Attack (M) 30,-- DM, Moon Patrol (M) 50,-- DM,Pyramid of Doom (C) 10,-- DM, Gohst Town (C) 10,-- DM,Assemblerkurs ASM-4 (D) 50,-- DM, Disassembler Masch. SP. (D) 50,-- DM.
Hardware: Super Sketch 150,-- DM.
Literatur: 99 Special 1 30,-- DM, 99 Special 2 30,-- DM, Extended B-- DM, Extended Basic D. Hasse Verlag 20,-- DM, TI99/4A Intern v. Heiner Martin 20,-- DM.
R. Lohmüller, Mühlhofstr. 14,7450 Hechingen- Stetten

*** R. G. SOFTGAME ***
Drehen von Sprites um die Hochachse!!!SUPER!!!
EX. B Spritecode eingeben und schon dreht sich dieser!
Grafik- Paket SPITZE!!
Mit diesem EX. B Programm können Sie sogar mit dem Joystick zeichnen!! + 4 SUPER- DEMOS.
Weitere Prg. bei Frank, Rückumschlag bei R. Goerke, Watzing 10, 8250 Dorfen

Verkaufe Konsole 175,-- DM, Rec.- Kabel 8,-- DM, 32- K extern 250,-- DM, Mini- Assembler 85,-- DM, 2 Joystick 50,-- DM, Extended Basic 2 plus 280,-- DM, donkey kong 40,-- DM,
ferner folgende Bücher: 21 listige Prg. für den TI99/4A 16,-- DM, Basic ganz einfach 9,-- DM, Spielen, Lernen, Arbeiten mit dem TI 12,-- DM, TI- Basic/ EX.- Basic für Anfänger und Fortgeschrittene 25,-- DM, Texas Instruments je Programm 10,-- DM.
Alle Preise + 2,-- DM für Porto u. Verpackung bei Wolfdietrich Meyer von Mo-Sa/19-21 Uhr unter 05421/1877

Komplette TI-Anlage zu verkaufen: TI99/4A-Konsole, Peripheriebox, Diskettenlaufwerk orig. TI incl. Controller, 32K-Speichererweiterung, Speechsynthesizer, Extended-Basic, TI-Writer, Schachmodul zu einem Gesamtpreis von 1400,-- DM
Klaus Glaßer, Rosenweg 2, 8400 Regensburg, Tel.: 0941/704172



VERKAUFE: TI99/4A, P-Box, Diskettenlaufwerk original TI + Controller, Speichererweiterung 32K, Sprachsynthesizer, X-Basic, TI-Writer (Textverarbeitung original TI) und Schach Modul zu einem Gesamtpreis von 1400,-- DM. Klaus Glaßer, Rosenweg 2, 8400 Regensburg, Tel.: 0941/704172

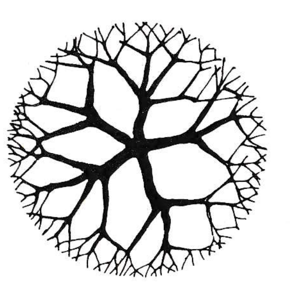
Verkaufe: 2*TI99/4A+Box+Diskcontroller+4Laufwerke+Diskmanager +32K+Centronics/2*RS232+E/A Modul+Extended Basic+Drucker GP 100A+Recorder+Monitor+TI-Invader+Joystick+viel Literatur und Zubehör. Heinz Bernemann, Vogelflug 9, 4796 Salzkotten 2, Tel.: 02955 1318, auch einzeln

VERKAUFE: Konsole TI99/4A+Erweiterungs-Box+Int. Disklaufwerk +Extended Basic-Modul+ 2 Spielmodule (TI-Invader+Hunt the Wumpus)+1 Modul Disc Manager 2: Verkaufspreis DM 1600,-- Noch dabei: 2 Joysticks (Feuertaste defekt), defekte 32K-Speichererweiterung seitl. ansteckbar, verschiedene Bücher bei: Leonhard Golinski 5120 Herzogenrath-Merkstein, Asternstr. 17, Tel.:0241/62250

Verkaufe neuwertige Module (deutsch): Statistik 40,-DM, Text/Datenverwaltung 60,-DM, Datenverwaltung/Analyse 40,-DM, Buchungsjournal 160,-DM, Centronics Schnittstelle direkt an Konsole, Bus durchgeführt, mit Kabel 160,-DM, DSK Rechnungsstellung 25,-DM. Tel.: (0531) 848884 oder 4722716 bei Rolf Kirchhoff, Lichtenberger Str.30, 3300 Braunschweig

Verkaufe Befehlserweiterungsmodul mit 8 neuen Befehlen (kein Ex.-Basic notwendig). Schaltung mit durchgeschliffenem Bus und Gehäuse für 30,-DM.
Verkaufe wegen Systemwechsel meine ganze Anlage möglichst komplett: TI99/4A Konsole & Extended Basic & Speichererweiterung (batteriegepuffert) & Kassettenrecorder & Literatur & Modul Othello & über 250,- Programme auf über 30 Kassetten (viele Maschienenprogramme) & Joysticks mit Adapter & alle TI-Revue Zeitschriften & div. andere Zeitschriften für 650,- DM Rainer Gawrikow, Sommerhausen 15, 5203 Much, 02245/3983





HARDWAREECKE

>>>>> DIE HARDWAREECKE <<<<<

Vom Clubmitglied Jörg Hirt stammt folgende interessante Erweiterung zu unserem TI. Mit der von Ihm entwickelten Schaltung ist es möglich, den TI von jedem Raum aus zu steuern. Durch ein kurzes Steuerprogramm können Anwendungs- und Spielprogramme geladen und gestartet werden. Ein Joystickanschluß an der Fernbedienung ist ebenfalls möglich.

Der Computer wird über den Pal Modulator und einen Antennenverstärker mit dem Fernsehapparat verbunden. Über ein 7-adriges Kabel (Abschirmung nicht unbedingt erforderlich) und einen 9-poligen Subminiaturstecker wird nun die Fernbedienung am Joystickport (J 300) des TI angeschlossen.

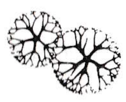
Die Tastatur besteht aus maximal 18 Tastenschaltern, die über das Steuerprogramm frei belegt werden können, und so jedes bestehende Programm angehängt werden kann. Die Schaltung wird nach dem beiliegenden Schaltbild mit max. 18 Schaltern und 13 Dioden (Typ 1N4148) auf einer Raster oder selbstgefertigten Platine aufgebaut.

Benötigtes Material:

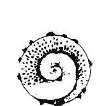
- 1 Rasterplatine
- 18 Tastenschalter z.B.
  IC Data-Tast, Digitast mini von Conrad Electronic Hirschau
  Original TI-Tastatur von Bühler Electronic
- 1 9 pol. Min D-Stiftleiste (Anschluß am Computer)
- 1 9 pol. Min D-Buchsenleiste (Anschluß für Joystick)
- 13 Dioden 1N4148
- 1 Gehäuse

Auf den nächsten Seiten folgt der Schaltplan und zwei Beispielprogramme zur Anwendung der Fernsteuerung. Das erste Beispiel demonstriert die Anwahl von verschiedenen Basicprogrammen über die Tastatur.

Das zweite Programm ist eine Subroutine, die in ein eigenes Basicprogramm eingebunden werden kann und die Fernsteuerung abfragt. Als Übergabeparameter wird die Tastenzahl angegeben. Weiteresinnvolle Anwendungen dieser Fernbedienung/Erweiterungstastatur könnt Ihr selber herausfinden (10er Tastenblock, Sonderzeichen, Graphiksteuerung etc.). Die Redaktion freut sich über Eure Vorschläge.





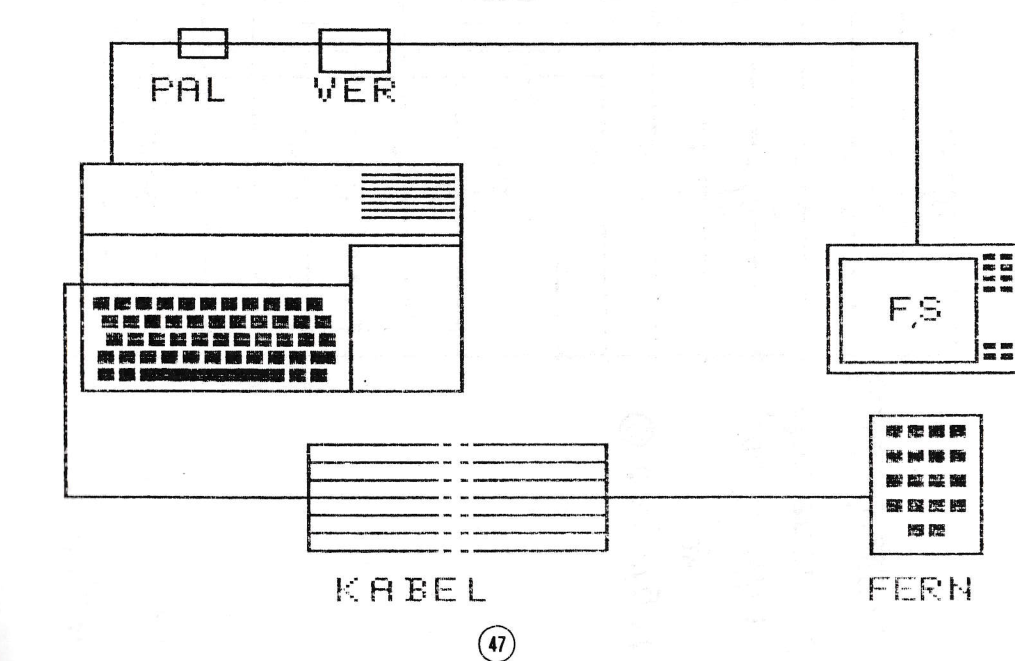


S C H A L T P L A N

TI 99/4A

F E R N B E D I E N U N G

COPYRIGHT BY JOERG H I R T



J300 Joystick Port

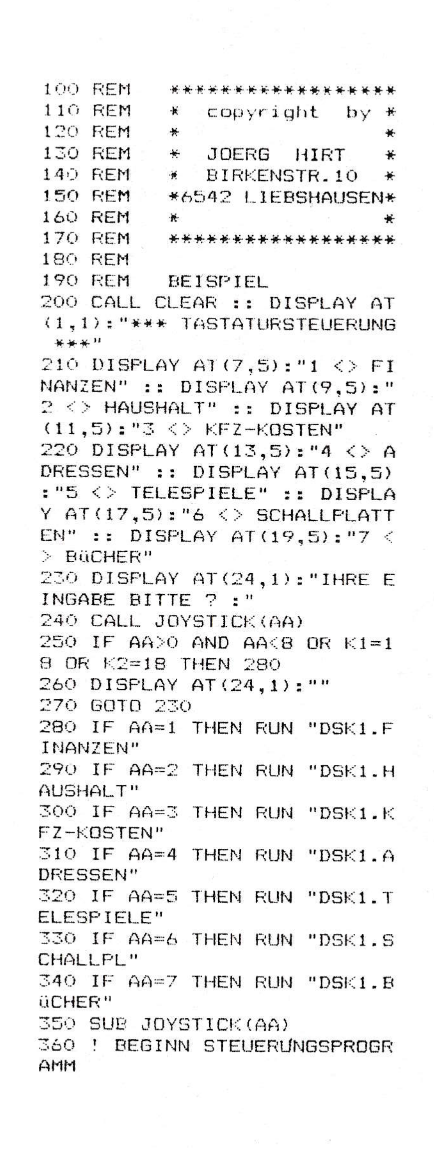
2 Joyst. B
7 Joyst. A
3 UP
9 RIGHT
8 DOWN
5 LEFT
4 PUSH BUTTON

TASTATUR

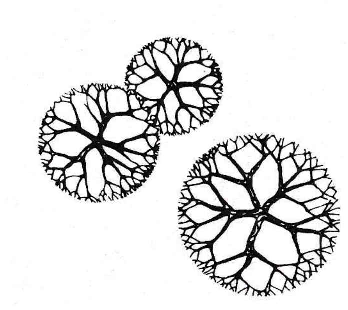
TI/99/4A

Joerg Hirt
Birkenstr.10
6542 Liebshausen

BEISPIEL ZUR FERNSTEUERUNG

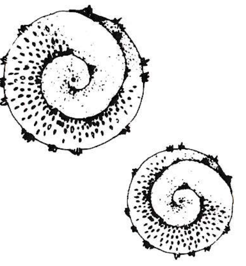


```
100 REM   *****************
110 REM   *  copyright  by *
120 REM   *                *
130 REM   *  JOERG  HIRT   *
140 REM   *  BIRKENSTR.10  *
150 REM   *6542 LIEBSHAUSEN*
160 REM   *                *
170 REM   *****************
180 REM
190 REM    BEISPIEL
200 CALL CLEAR :: DISPLAY AT
(1,1):"*** TASTATURSTEUERUNG
 ***"
210 DISPLAY AT(7,5):"1 <> FI
NANZEN" :: DISPLAY AT(9,5):"
2 <> HAUSHALT" :: DISPLAY AT
(11,5):"3 <> KFZ-KOSTEN"
220 DISPLAY AT(13,5):"4 <> A
DRESSEN" :: DISPLAY AT(15,5)
:"5 <> TELESPIELE" :: DISPLA
Y AT(17,5):"6 <> SCHALLPLATT
EN" :: DISPLAY AT(19,5):"7 <
> BüCHER"
230 DISPLAY AT(24,1):"IHRE E
INGABE BITTE ? :"
240 CALL JOYSTICK(AA)
250 IF AA>0 AND AA<8 OR K1=1
8 OR K2=18 THEN 280
260 DISPLAY AT(24,1):""
270 GOTO 230
280 IF AA=1 THEN RUN "DSK1.F
INANZEN"
290 IF AA=2 THEN RUN "DSK1.H
AUSHALT"
300 IF AA=3 THEN RUN "DSK1.K
FZ-KOSTEN"
310 IF AA=4 THEN RUN "DSK1.A
DRESSEN"
320 IF AA=5 THEN RUN "DSK1.T
ELESPIELE"
330 IF AA=6 THEN RUN "DSK1.S
CHALLPL"
340 IF AA=7 THEN RUN "DSK1.B
üCHER"
350 SUB JOYSTICK(AA)
360 ! BEGINN STEUERUNGSPROGR
AMM
```



STEUERUNGSPROGRAMM ZUR FERNSTEUERUNG

```
100 REM    ******************
110 REM    *  copyright  by *
120 REM    *                *
130 REM    *  JOERG  HIRT   *
140 REM    *  BIRKENSTR.10  *
150 REM    *6542 LIEBSHAUSEN*
160 REM    *                *
170 REM    ******************
180 REM
190 REM    TASTATURSTEUERUNG
200 SUB JOYSTICK(AA)
210 AA=0 :: DISPLAY AT(24,27
):"**" :: CALL JOYST(1,X1,Y1
):: CALL JOYST(2,X2,Y2):: CA
LL KEY(1,K1,S1):: CALL KEY(2
,K2,S2)
220 IF X1<>0 OR Y1<>0 OR K1=
18 THEN 240
230 IF X2<>0 OR Y2<>0 OR K2=
18 THEN 260 ELSE SUBEXIT
240 DISPLAY AT(24,27):"" ::
IF K1=18 THEN AA=17 :: GOTO
280 ELSE IF X1=0 AND Y1=4 TH
EN AA=1 :: GOTO 280 ELSE IF
X1=4 AND Y1=4 THEN AA=2 :: G
OTO 280 ELSE IF X1=4 AND Y1=
0 THEN AA=3 :: GOTO 280 ELSE
 IF X1=4 AND Y1=-4 THEN AA=4
 :: GOTO 280
250 IF X1=0 AND Y1=-4 THEN A
A=5 :: GOTO 280 ELSE IF X1=-
4 AND Y1=-4 THEN AA=6 :: GOT
O 280 ELSE IF X1=-4 AND Y1=0
 THEN AA=7 :: GOTO 280 ELSE
IF X1=-4 AND Y1=4 THEN AA=8
:: GOTO 280
260 DISPLAY AT(24,27):"" ::
IF K2=18 THEN AA=18 :: GOTO
280 ELSE IF X2=0 AND Y2=4 TH
EN AA=9 :: GOTO 280 ELSE IF
X2=4 AND Y2=4 THEN AA=10 ::
GOTO 280 ELSE IF X2=4 AND Y2
=0 THEN AA=11 :: GOTO 280 EL
SE IF X2=4 AND Y2=-4 THEN AA
=12 :: GOTO 280
270 IF X2=0 AND Y2=-4 THEN A
A=13 :: GOTO 280 ELSE IF X2=
-4 AND Y2=-4 THEN AA=14 :: G
OTO 280 ELSE IF X2=-4 AND Y2
=0 THEN AA=15 :: GOTO 280 EL
SE IF X2=-4 AND Y2=4 THEN AA
=16
280 SUBEND
```



*** Softwareprodukte der Firma Power Soft ***

Von der Firma Power Soft wurden der Redaktion vier Spiele zum Testen überlassen: Heint, Sommerspiele, Flugsimulator (Nonstop) und Warlock.

Alle Spiele zeichnen sich durch eine gute, aufwendig gestaltete Grafik aus. Der Programmspeicher wird fast immer voll ausgenutzt. Jedoch ist der Spielablauf in der Regel recht langsam, was natürlich an den begrenzten Möglichkeiten des Bascis liegt. Aber es fehlt eine trickreiche Programmierung, mit der doch so einiges am Spielverlauf beschleunigt, bzw. interessanter gemacht werden könnte.

Am meisten Action bieten die Sommerspiele. Dabei treten Sie im 100m Lauf, Weitsprung und Speerwerfen an. Das Programm ist für max. 4 Mitspieler ausgelegt. Wenn man mit mehreren zusammen spielt, kommt schnell eine richtige Wettkampfstimmung auf.

Der Flugsimulator ist von einem geduldigen Spieler schnell in den Griff zu bekommen, und schon bald gelingen Start und Landung. Es können verschiedene Schwierigkeitsgrade gewählt werden, wodurch das Spiel auch noch länger interessant bleibt.

Die anderen Spiele sind mehr Abenteuerspiele. An Action recht arm, aber voller Überraschungen. Wenn man bis zur Lösung kommen will, vergehen so manche Nachmittage.

Wer also Spaß an der Grafik hat, und wen eklatante Schreibfehler in den Anleitungen nicht stören, der bekommt für sein Geld eine unterhaltsame Software.

IMPRESSUM

Herausgeber und Chefredakteur:
Hartmut Dirks
2. Chefredakteur:
Gregor Lohmann
Verantwortlich für den Inhalt:
Für den allgemeinen Teil
Hartmut Dirks.
Für Ihre Artikel und Programme
die einzelnen Autoren.
Mitarbeiter dieser Ausgabe:
H. Reidlinger, K. Hagenbuchner,
Hans-Joachim Gerstein, Roman
Romanowski, Wolfgang Enders,
Jörg Hirt, Michael Eckert,
Rainer Gawrikow.
Layout:
Roswitha Schultz
Private Inserate sind kostenlos.
Kommerzielle Anzeigenpreise gegen
Anfrage.
Nachdruck verboten.
Es besteht kein Rechtsanspruch auf
Lieferung.
Redaktionsadresse:
TI99/4A-REPORT
Hartmut Dirks
Reutershagweg 21
5100 Aachen
Telefon: 0241 - 87 22 05
Bankverbindung:
Stadtsparkasse Aachen
Bankleitzahl: (390 500 00)
Konto Nr.: 22 00 23 23